

Atividade Problemática

1 Introdução

A ferramenta [Dash](#) foi criada com base na ferramenta [Plotly](#), que é uma biblioteca poderosa em Python para gerar gráficos dinâmicos. Com ela, pode-se dar *zoom*, clicar, escolher uma faixa de valores, ver os valores, entre outras funcionalidades. Essa biblioteca foi utilizada como base para criar *dashboards*.

Esses *dashboards* podem ser extremamente ricos, compostos por diversos gráficos, gráficos dinâmicos, tabelas onde conseguimos fazer filtragens, ou até mesmo formulários com os quais é possível interagir.

Neste roteiro será implementado um *dashboard* que possui três páginas:

- Uma página inicial, apenas de boas-vindas
- Uma página apresentando dois tipos de gráficos diferentes para que se aprenda como criar um gráfico do Plotly e colocá-lo dentro de um *dashboard*
- Um formulário onde será feita uma previsão, de acordo com o modelo treinado

Além disso, será preciso lidar com diversas questões de *front-end*, porque quando fala em mostrar um *dashboard*, as questões que surgem não são apenas do *back-end*, de trazer os dados para montar o gráfico, mas também de como apresentar isso na nossa página.

Isso envolve decidir se um gráfico ficará ao lado do outro, um em cima do outro, qual será o tamanho do *grid*, a proporção de um para o outro, o que acontece em uma tela pequena e em uma tela grande, entre outras questões de componentes visuais apresentados para nossos clientes dentro de um *dashboard*.

O código do *back-end* será feito em Python, ele irá trabalhar com os dados e produzir os gráficos. É extremamente importante entender cada vez mais sobre *front-end*, pois isso trará consequências para nossos *dashboards*.

Será usado um modelo já treinado de previsão de doença cardíaca e, de acordo com os dados que preenchidos no formulário, o modelo será executado e fornecerá sua previsão.

É um modelo simples, mas com dados do mundo real, utilizados de uma universidade americana, a UCI. Estes dados serão utilizados para criar um modelo, produzir o formulário e, de acordo com o modelo, executar para esses dados e obter uma previsão.

Dash apresenta três pilares tecnológicos essenciais:

- [Flask](#): *Micro-framework* para desenvolvimento web em Python. Quando é criado um *dashboard* com o Dash, por baixo dos panos, ela está utilizando o Flask para lidar com as comunicações entre o navegador do usuário (cliente) e o servidor onde o aplicativo está rodando. Isso inclui receber pedidos do navegador, como quando interagimos com um componente da Dash, e enviar as respostas apropriadas do servidor de volta ao navegador, permitindo assim a interatividade e a atualização dos componentes do aplicativo.
- [Plotly.js](#): Biblioteca de gráficos JavaScript de código aberto que a Dash utiliza para renderizar visualizações de dados interativas.
- [React](#): Biblioteca JavaScript para construir interfaces de usuário. Dash utiliza *React.js* para renderizar os componentes do lado do cliente. Cada componente da Dash é essencialmente um componente React que pode ser interativo e dinâmico.

2 IDE

Pode ser utilizada qualquer IDE ou editor de código para este trabalho. Caso seja usado o Spyder (IDE *default* do Anaconda), é importante **NÃO** executar o programa de dentro da IDE. Neste caso, o programa deve ser executado a partir do *prompt* de comando de dentro do ambiente virtual a ser criado nas próximas seções.

3 Dataset

Dataset de 1988 da UCI (Universidade da Califórnia, Irvine) sobre doenças cardíacas:

The screenshot shows the "Heart Disease" dataset page on the UC Irvine Machine Learning Repository. The page includes a summary table, dataset characteristics, feature types, dataset information, additional information, missing values, and creator details.

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Health and Medicine	Classification
Feature Type	# Instances	# Features
Categorical, Integer, Real	303	13

Dataset Information

Additional Information: This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have ...

Show More ▾

Has Missing Values? Yes

Creators

- Andras Janosi
- William Steinbrunn
- Matthias Pfisterer
- Robert Detrano

DOI: 10.24432/C52P4X

4 Instalando o Repositório e Importando os Dados

Crie um *notebook* no Google Colab e instale o pacote contendo o repositório de doenças cardíacas da UCI:

```
➤ !pip3 install -U ucimlrepo
```

A seguir importe o *dataset* usando o código fornecido pela UCI (botão *Import in Python*):

The screenshot shows the "Import the dataset into your code" section of the "Heart Disease" dataset page. It displays a code snippet for importing the dataset using the ucimlrepo package.

```
from ucimlrepo import fetch_ucirepo  
  
# fetch dataset  
heart_disease = fetch_ucirepo(id=45)  
  
# data (as pandas dataframes)  
X = heart_disease.data.features  
y = heart_disease.data.targets  
  
# metadata  
print(heart_disease.metadata)  
  
# variable information  
print(heart_disease.variables)
```

Apague estas 3 linhas (elas serão refeitas mais à frente):

```
# data (as pandas dataframes)  
X = heart_disease.data.features  
y = heart_disease.data.targets
```

Ao executar este código ele deve mostrar os metadados do *dataset* e também a descrição das variáveis:

```
{'uci_id': 45, 'name': 'Heart Disease', 'repository_url': 'https://archive.ics.uci.edu/dataset/45/heart+disease', 'data_url':  
    name      role      type demographic \  
0     age     Feature   Integer    Age  
1     sex     Feature   Categorical  Sex  
2     cp     Feature   Categorical  None  
3   trestbps Feature   Integer    None  
4     chol    Feature   Integer    None  
5     fbs     Feature   Categorical  None  
6   restecg Feature   Categorical  None  
7   thalach Feature   Integer    None  
8     exang   Feature   Categorical  None  
9   oldpeak Feature   Integer    None  
10    slope   Feature   Categorical  None  
11    ca      Feature   Integer    None  
12   thal     Feature   Categorical  None  
13     num     Target    Integer    None  
  
                                         description  units missing_values  
0                           None    years      no  
1                           None    None       no  
2                           None    None       no  
3 resting blood pressure (on admission to the ho... mm Hg      no  
4                               serum cholesterol mg/dl      no  
5                               fasting blood sugar > 120 mg/dl  None      no  
6                               None    None       no  
7                               maximum heart rate achieved  None      no  
8                               exercise induced angina  None      no  
9 ST depression induced by exercise relative to ...  None      no  
10                          None    None       no  
11 number of major vessels (0-3) colored by flour...  None      yes  
12                          None    None       yes  
13 diagnosis of heart disease  None      no
```

5 Descrição dos Dados

Inicialmente, o conjunto de dados continha 76 variáveis, mas todas as análises realizadas concentram-se no uso de um subconjunto de 14 delas. Vale destacar que, até o momento, o banco de dados da *Cleveland Clinic* é o único utilizado por pesquisadores de aprendizado de máquina. As análises realizadas com esse banco de dados têm como objetivo distinguir a presença da doença cardíaca (valores 1, 2, 3, 4) da sua ausência (valor 0).

Neste projeto os valores maiores que zero serão substituídos por 1, o que significa que um valor de 1 indica a presença de doença cardíaca, enquanto um valor de 0 indica a ausência da mesma.

Saber quais são as *features* do *dataset* é fundamental para compreender mais a fundo as informações e os padrões que podem indicar a presença ou ausência de doença cardíaca.

A seguir uma explicação sobre cada uma delas:

- **age:** Idade em anos;
- **sex:** Sexo biológico (0 = feminino, 1 = masculino);
- **cp:** Tipo de dor no peito relatada pelo paciente (1= angina típica, 2 = angina atípica, 3 = não angina, 4 = angina assintomática);

Angina é uma condição médica caracterizada por dor ou desconforto no peito.

- **trestbps:** Pressão arterial medida em repouso;

Pressão arterial é a força que o sangue exerce contra as paredes das artérias à medida que é bombeado pelo coração para o resto do corpo.

- **chol:** Nível de colesterol no sangue em miligramas por decilitro (mg/dl);
- **fbs:** Nível de glicose no sangue em jejum, indicando se está abaixo de 120 mg/dl ou acima de 120 mg/dl (0 = abaixo, 1 = acima);
- **restecg:** Resultados do eletrocardiograma em repouso (0 = normal, 1 = anormalidade de onda ST-T, 2 = hipertrofia ventricular esquerda);

O **eletrocardiograma**, frequentemente abreviado como ECG, é um exame médico que registra a atividade elétrica do coração ao longo do tempo. Quando os resultados da eletrocardiografia (ECG) em repouso são considerados "normais", isso significa que a atividade elétrica do coração está dentro dos padrões regulares. Já quando existe anormalidade da onda ST-T pode ser um indicativo de algum tipo de irregularidade na função cardíaca. Por fim, na hipertrofia ventricular esquerda temos um aumento do tamanho do músculo cardíaco no lado esquerdo do coração.

- **thalach:** Frequência cardíaca máxima alcançada durante um teste de esforço físico;
- **exang:** Indica se houve angina (dor no peito) induzida por exercício ou não (0 = sim, 1 = não);

- **oldpeak:** Medida da depressão do segmento ST induzida pelo exercício em relação ao repouso; Vamos imaginar o coração como uma bomba, e o eletrocardiograma (ECG) como um gráfico que mostra como essa bomba está funcionando. O **segmento ST** é como um intervalo na leitura desse gráfico que diz quando o coração está relaxando depois de bater. Agora, se durante esse relaxamento, o gráfico mostra uma parte chamada "segmento ST" mais baixa do que o normal, é como se o coração dissesse "Ei, não estou recebendo sangue suficiente aqui!" Essa baixa no gráfico é chamada de **depressão do segmento ST**.
- **slope:** Inclinação do segmento ST no pico do exercício (1 = inclinado p/cima, 2 = plano, 3 = inclinado p/baixo); Os resultados basicamente descrevem como o coração responde ao esforço físico, olhando para um gráfico do batimento cardíaco e vendo se ele sobe, fica nivelado ou desce nesse momento específico do exercício.
- **ca:** Número de vasos sanguíneos principais coloridos durante o procedimento de fluoroscopia; A fluoroscopia dos vasos sanguíneos é um exame de imagem para visualizar em tempo real o fluxo de sangue nos vasos sanguíneos.
- **thal:** Resultado do exame de cintilografia com tálio (3 = normal, 6 = defeito fixo, 7 = defeito reversível); A cintilografia com tálio é um procedimento de imagem que emprega uma substância radioativa para analisar o fluxo sanguíneo e a função cardíaca. Esse exame é valioso para detectar regiões do coração que apresentam comprometimento ou baixa viabilidade.

6 Análise Exploratória dos Dados

Crie uma variável `dados` (`DataFrame Pandas`) contendo um subconjunto de 13 *features* (atributos preditores) e mostre os primeiros registros:

```
> dados = heart_disease.data.features
> dados.head()
```

Plote o histograma (com 30 bins) das idades a fim de fazer uma análise exploratória inicial:

```
> import matplotlib.pyplot as plt
> plt.hist(dados['age'], bins=30, edgecolor='black')
> plt.xlabel('Idade')
> plt.ylabel('Frequência')
> plt.title('Distribuição Etária')
> plt.show()
```

A variável 58 (`num`) é o atributo *target* (alvo) contém o diagnóstico, ou seja, se a pessoa tem ou não doença cardíaca. Valores maiores ou iguais a 1 significam doença e 0 não tem doença.

Para ver os valores disponíveis, execute o seguinte:

```
> heart_disease.data.targets
```

A linha abaixo mostra `True` em caso de doença e `False` caso contrário:

```
> heart_disease.data.targets > 0
```

Para produzir uma classificação binária, '0: Não tem doença' e '1: Com doença', pode-se usar o seguinte truque usando multiplicação:

```
> 1 * (heart_disease.data.targets > 0)
```

Criando uma coluna extra no `DataFrame` contendo essa classificação:

```
> dados['doenca'] = 1 * (heart_disease.data.targets > 0)
> dados.head()
```

Resultado:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	doenca
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0

Visualizando o *boxplot* com o `seaborn`:

```
> import seaborn as sns
> sns.boxplot(y='age', x='doenca', data=dados, hue='doenca')
> plt.title('Distribuição Etária por Doença')
> plt.xlabel('Diagnóstico (0 = não | 1 = sim)')
> plt.ylabel('Idade')
> plt.show()
```

7 Criação de Gráfico Interativo com Plotly Express

Código:

```
> import plotly.express as px
> fig = px.histogram(dados, x='age', nbins=30, title='Distribuição Etária (Plotly)',
>                      color='doenca')
> fig.update_layout(xaxis_title='Idade', yaxis_title='Frequência')
> fig.show()
```

Exercício: Usando o Plotly, crie dois outros *boxplots*, um deles de ‘Colesterol sérico vs Doença’ e outro de ‘Pressão arterial em repouso vs Doença’.

8 Criação do Ambiente Virtual Local

A partir deste ponto o roteiro assume que o acadêmico instalou a distribuição [Anaconda](#).

No seu computador, crie uma pasta para o projeto, por exemplo, *c:\programas\ml*.

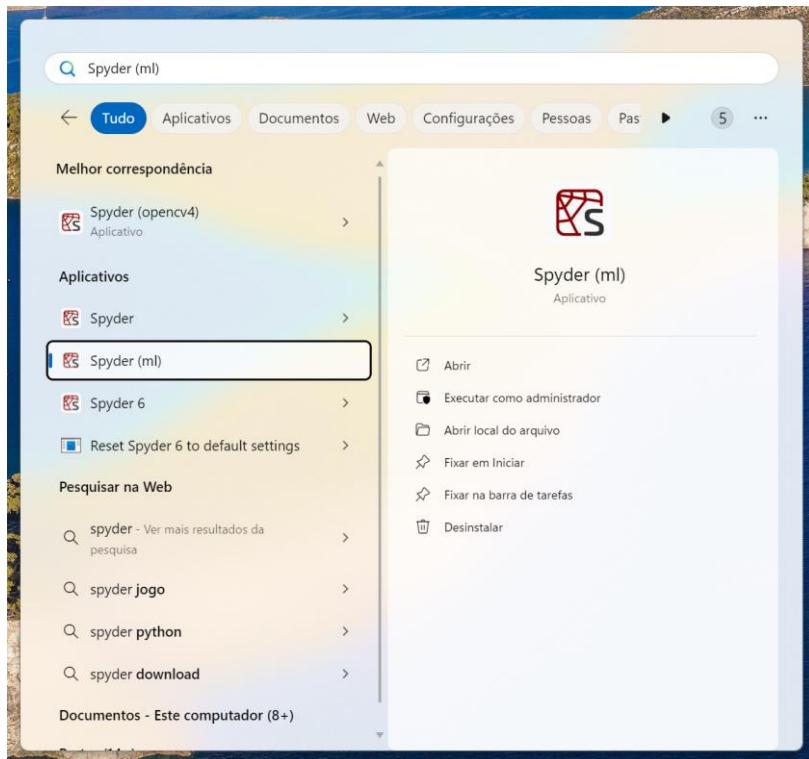
Dentro desta pasta crie um arquivo chamado *requirements.txt* com o seguinte conteúdo:

```
plotly==5.24.1
ucimlrepo==0.0.7
pandas==2.2.3
dash==3.2.0
dash-bootstrap-components==2.0.3
dash-bootstrap-templates==1.3.0
scikit-learn==1.6.1
xgboost==3.0.4
```

Para criar um ambiente virtual e instalar as bibliotecas necessárias, abra o Anaconda *Prompt* e execute os seguintes comandos:

```
> conda create --name ml -c conda-forge python=3.13.5 spyder=6.0.7 numpy scipy pandas matplotlib
sympy cython
> conda activate ml
> cd /d c:\programas\ml
> pip3 install -r requirements.txt
```

Com isso agora deverá aparecer o Spyder para ambiente virtual *ml* (outra IDE pode ser usada, como o VS Code):



Após terminar o trabalho, se não quiser mais manter o ambiente virtual, ele pode ser removido da seguinte forma:

```
> conda deactivate
> conda env remove --name ml
```

9 Criação do Aplicativo e Funcionalidade de Visualização de Gráficos

9.1 Aplicativo Python

No Spyder, crie o arquivo *app_graficos.py* dentro da pasta *c:\programas\ml* e insira nele o código a seguir:

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo
import plotly.express as px

# Importa os dados
heart_disease = fetch_ucirepo(id=45)
dados = heart_disease.data.features

# Acrescentando a coluna 'doenca'
dados['doenca'] = 1 * (heart_disease.data.targets > 0)

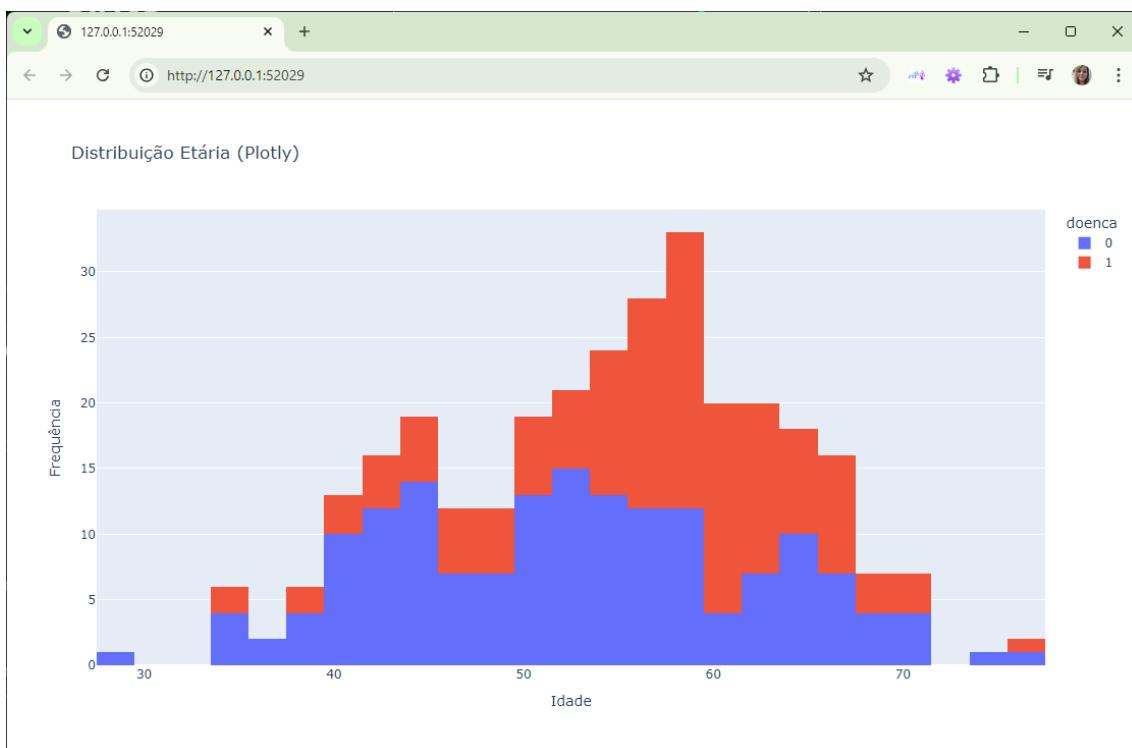
# Análise exploratória
print(dados.head())

# Cria o histograma
fig_hist = px.histogram(dados, x='age', nbins=30, title='Distribuição Etária (Plotly)',
color='doenca')
fig_hist.update_layout(xaxis_title='Idade', yaxis_title='Frequência')
fig_hist.show()
```

Salve o arquivo e, de dentro do Anaconda *Prompt*, execute o programa:

```
➤ python app_graficos.py
```

Será aberta uma página no navegador mostrando o histograma:



Será mostrado também as primeiras linhas do *dataset* no *console*.

```
(ml) C:\programas\ml>python app.py
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  doenca
0   63    1    1      145   233    1  ...     0      2.3     3  0.0   6.0      0
1   67    1    4      160   286    0  ...     1      1.5     2  3.0   3.0      1
2   67    1    4      120   229    0  ...     1      2.6     2  2.0   7.0      1
3   37    1    3      130   250    0  ...     0      3.5     3  0.0   3.0      0
4   41    0    2      130   204    0  ...     0      1.4     1  0.0   3.0      0

[5 rows x 14 columns]
(ml) C:\programas\ml>
```

Modifique o programa acrescentando/modificando as linhas em amarelo a fim de instanciar a aplicação:

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo
import plotly.express as px
from dash import Dash, dcc, html

# Importa os dados
heart_disease = fetch_ucirepo(id=45)
dados = heart_disease.data.features

# Acrescentando a coluna 'doenca'
dados['doenca'] = 1 * (heart_disease.data.targets > 0)

# Análise exploratória
#print(dados.head())

# Cria o histograma
fig_hist = px.histogram(dados, x='age', nbins=30, title='Distribuição Etária (Plotly)',
color='doenca')
fig_hist.update_layout(xaxis_title='Idade', yaxis_title='Frequência')
#fig_hist.show()

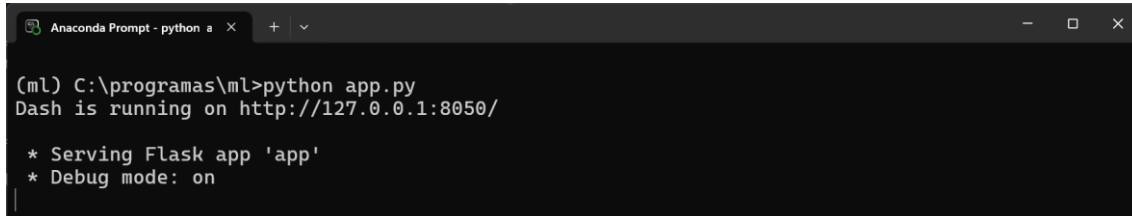
# Instancia o aplicativo
app = Dash(__name__)
app.layout = html.Div([
    html.H1('Análise de Dados do UCI Repository Heart Disease'),
    dcc.Graph(figure=fig_hist)
])

app.run(debug=True)
```

A função do `app.layout` é definir a estrutura e os componentes visuais da página web, organizando-os em uma disposição específica para a interface do usuário. Já habilitar o modo de depuração na última linha do código permite que o desenvolvedor veja as alterações em tempo real e receba *feedback* imediato sobre erros, o que facilita a correção e a otimização do código.

Execute a aplicação no *prompt*:

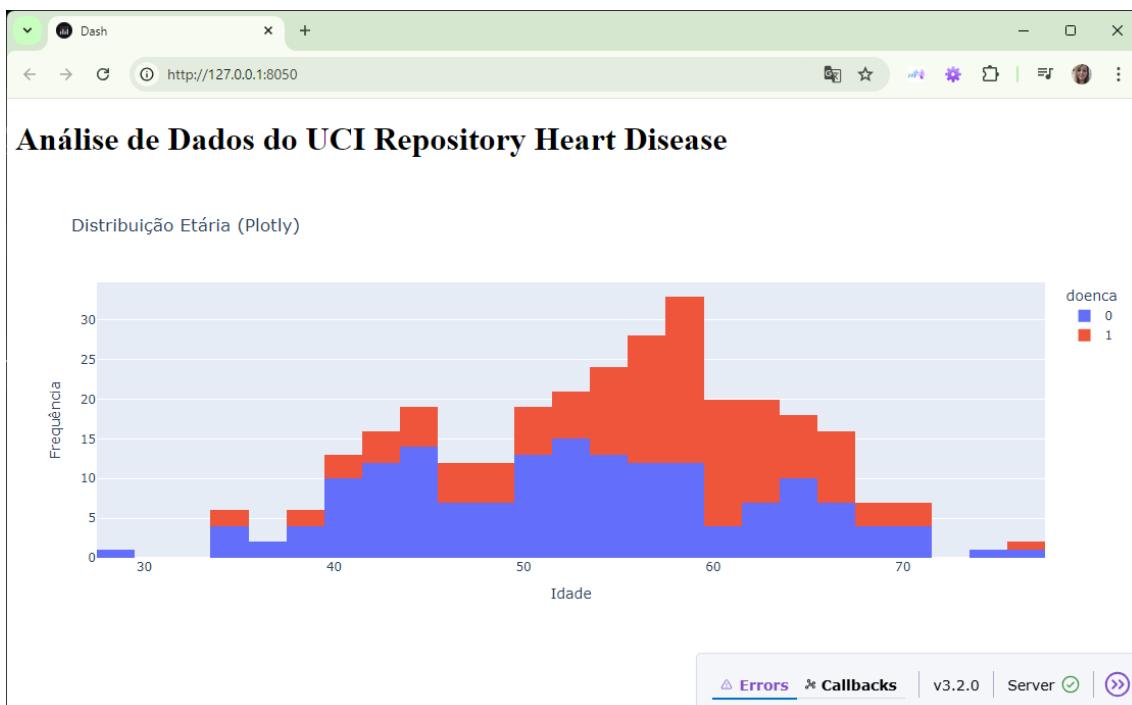
```
➤ python main.py
```



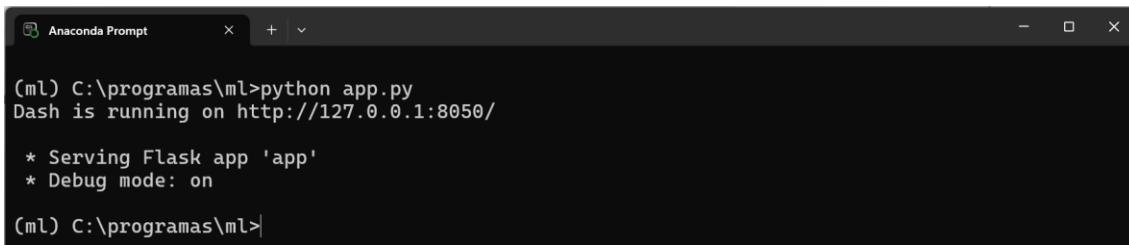
```
Anaconda Prompt - python a x + v

(ml) C:\programas\ml>python app.py
Dash is running on http://127.0.0.1:8050/
* Serving Flask app 'app'
* Debug mode: on
```

Abra a seguinte URL no navegador: <http://127.0.0.1:8050/>



Para cancelar a execução, pressione *Ctrl + C* na tela de *prompt*:



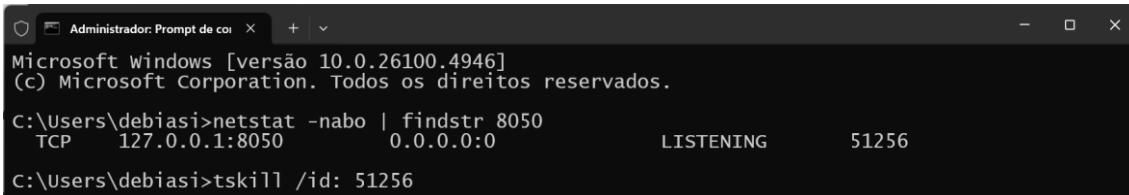
```
(ml) C:\programas\ml>python app.py
Dash is running on http://127.0.0.1:8050/
* Serving Flask app 'app'
* Debug mode: on
(ml) C:\programas\ml>
```

Caso você tenha modificado o código-fonte, executado novamente mas a página não tenha alterado, possivelmente é devido a alguma instância anterior que ainda está na memória. Para interromper o processo antigo, abra um *prompt* elevado (em modo administrador) e execute o comando abaixo, que mostra as portas 8050 abertas:

```
➤ netstat -nabo | findstr 8050
➤ tskill /id: [código do processo]
```

Neste caso o processo é o 51256 (mostrado ao lado da palavra LISTENING – OUVINDO):

```
➤ tskill /id: 51256
```



```
Administrator: Prompt de cor > Microsoft Windows [versão 10.0.26100.4946]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\debiasi>netstat -nabo | findstr 8050
    TCP    127.0.0.1:8050        0.0.0.0:0          LISTENING      51256
C:\Users\debiasi>tskill /id: 51256
```

Modifique o programa e acrescente o gráfico *boxplot*:

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo
import plotly.express as px
from dash import Dash, dcc, html

# Importa os dados
heart_disease = fetch_ucirepo(id=45)
dados = heart_disease.data.features

# Acrescentando a coluna 'doenca'
dados['doenca'] = 1 * (heart_disease.data.targets > 0)

# Análise exploratória
#print(dados.head())

# Cria o histograma
fig_hist = px.histogram(dados, x='age', nbins=30, title='Distribuição Etária (Plotly)',
color='doenca')
fig_hist.update_layout(xaxis_title='Idade', yaxis_title='Frequência')

# Cria o boxplot
fig_box = px.box(dados, x='doenca', y='age', title='Distribuição Etária por Doença', color='doenca')
fig_box.update_layout(xaxis_title='Diagnóstico (0 = não | 1 = sim)', yaxis_title='Idade')

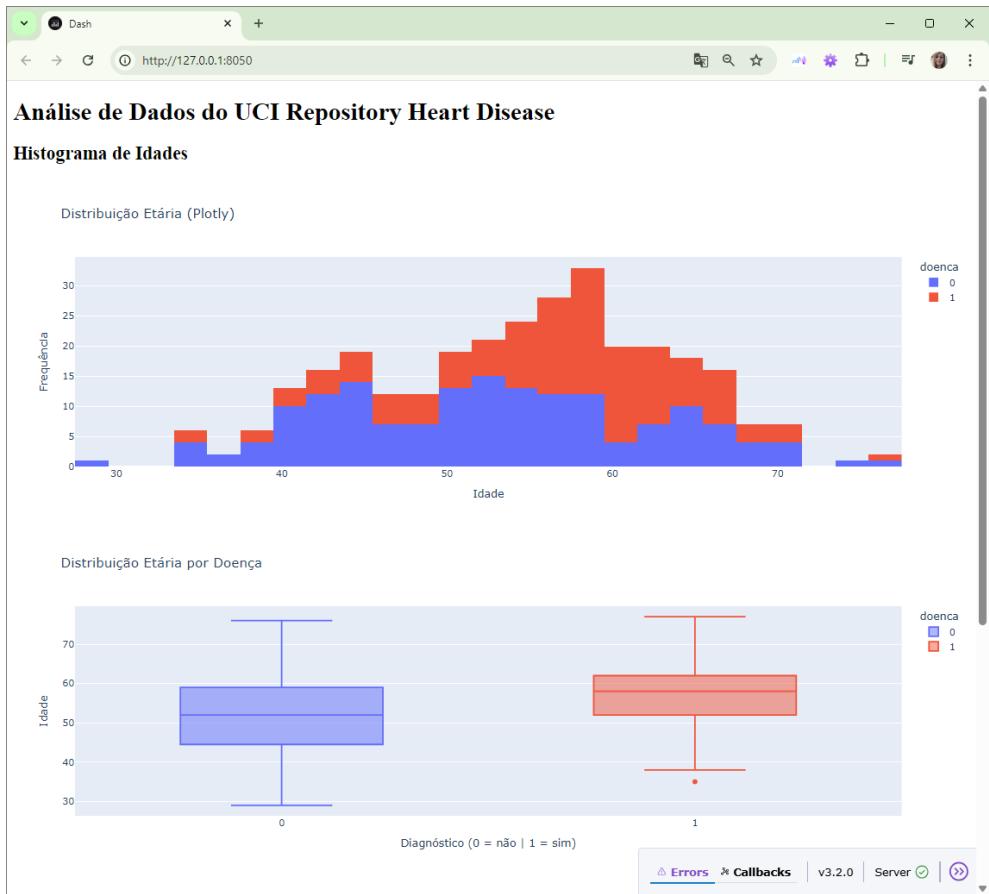
# Cria as divs
div_hist = html.Div([
    html.H2('Histograma de Idades'),
    dcc.Graph(figure=fig_hist),
])

div_box = html.Div([
    html.H2('Boxplot de Idades'),
    dcc.Graph(figure=fig_box)
])

# Instancia o aplicativo
app = Dash(__name__)
app.layout = html.Div([
    html.H1('Análise de Dados do UCI Repository Heart Disease'),
    div_hist,
    div_box
])

app.run(debug=True)
```

Salve e execute novamente o programa:



9.2 Componentes HTML

É fundamental ter um entendimento sólido dos elementos HTML que compõem a estrutura de qualquer página ou aplicação *web*. Os elementos HTML são as unidades básicas de construção que nos permitem criar conteúdo e definir a estrutura e o *layout* de uma página *web*.

Alguns dos elementos HTML disponíveis no módulo `html` da biblioteca Dash:

- Elemento `Div`: É um dos elementos mais utilizados no HTML para a organização do *layout* de uma página. Ele é um contêiner genérico para o conteúdo de fluxo, que não representa nada por si só. No entanto, juntamente com CSS (*Cascading Style Sheets*), o `Div` é extremamente útil para estilizar blocos de conteúdo, organizar seções da página e desenvolver *layouts* complexos através de técnicas como *flexbox* e *grid*. O uso do `Div` como elemento principal em `app.layout` ilustra como os elementos podem ser aninhados dentro de um `Div` para criar uma estrutura lógica e visualmente organizada.
- `html.H1`: Usado para inserir um cabeçalho de nível superior na página. É o título de nível mais importante e geralmente é usado para o título principal da aplicação ou da página. Ele ajuda na organização do conteúdo e na melhoria da acessibilidade e SEO (*Search Engine Optimization*) do aplicativo.
- `html.H2`: Similar ao `html.H1`, mas usado para subcabeçalhos ou títulos de seções secundárias.
- `html.Label`: É utilizado para adicionar etiquetas a elementos de entrada de dados, como caixas de texto, *radio buttons*, e *checkboxes*. As etiquetas são importantes para a acessibilidade, pois fornecem descrições claras dos campos de entrada para os usuários, incluindo aqueles que utilizam leitores de tela.
- `html.Button`: Cria um botão no *layout* do aplicativo permitindo interações dos usuários como executar ações como enviar formulários, filtrar dados ou qualquer outra que o desenvolvedor deseje implementar. A personalização através de CSS pode alterar sua aparência para se adequar ao *design* do aplicativo.
- `html.P`: Representa um parágrafo de texto. É utilizado para adicionar conteúdo textual ao aplicativo, como descrições, instruções ou qualquer informação relevante. Parágrafos ajudam a quebrar o texto em blocos mais digestíveis, melhorando a legibilidade e a experiência do usuário.

10 Treinamento do Modelo

Dentro do projeto, crie o arquivo *treina_modelo.py* com o código abaixo:

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo

# Importa os dados
heart_disease = fetch_ucirepo(id=45)
dados = heart_disease.data.features

# Acrescentando a coluna 'doenca'
dados['doenca'] = 1 * (heart_disease.data.targets > 0)

# Separando os atributos preditores do atributo alvo (target)
X = dados.drop(columns='doenca')
y = dados['doenca']

print('Atributos preditores:\n', X.head())
print('\nTarget (alvo):\n', y.head())
```

Execute o arquivo *treina_modelo.py*:

```
> python treina_modelo.py
```

```
Anaconda Prompt
(ml) C:\programas\ml>python treina_modelo.py
Atributos preditores:
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal
0    63    1    1      145   233    1      2     150      0     2.3    3  0.0   6.0
1    67    1    4      160   286    0      2     108      1     1.5    2  3.0   3.0
2    67    1    4      120   229    0      2     129      1     2.6    2  2.0   7.0
3    37    1    3      130   250    0      0     187      0     3.5    3  0.0   3.0
4    41    0    2      130   204    0      2     172      0     1.4    1  0.0   3.0

Target (alvo):
  0
  1
  2
  0
  0
Name: doenca, dtype: int64
(ml) C:\programas\ml>
```

Comente as duas últimas linhas e acrescente as seguintes no arquivo *treina_modelo.py*:

```
# Separação dos dados entre treino e teste
from sklearn.model_selection import train_test_split

# 20% do dataset são separados para treino
# Parâmetro stratify: Garante que os conjuntos de treinamento e teste tenham a mesma proporção
# de classes (ou rótulos) que o conjunto de dados original
# Isso é particularmente útil ao lidar com conjuntos de dados desbalanceados, onde algumas
# classes são sub-representadas.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Biblioteca xgb
import xgboost as xgb

# XGBoost é um algoritmo de aprendizado de máquina que pertence à categoria de aprendizado
# por conjunto (ensemble), especificamente à estrutura de reforço de gradiente
# Ele utiliza árvores de decisão como aprendizes base e emprega técnicas de regularização
# para aprimorar a generalização do modelo
modelo = xgb.XGBClassifier(objective='binary:logistic')

# Treina o modelo usando o subconjunto de treinamento
modelo.fit(X_train, y_train)

# Faz previsões usando os dados de teste
preds = modelo.predict(X_test)

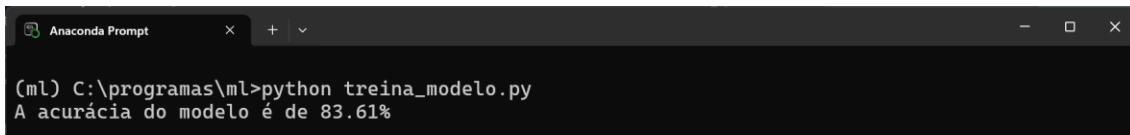
# Avaliação do modelo
from sklearn.metrics import accuracy_score

acuracia = accuracy_score(y_test, preds)
print(f'A acurácia do modelo é de {acuracia:.2%}')

# Biblioteca com várias funções úteis para processamento paralelo
import joblib

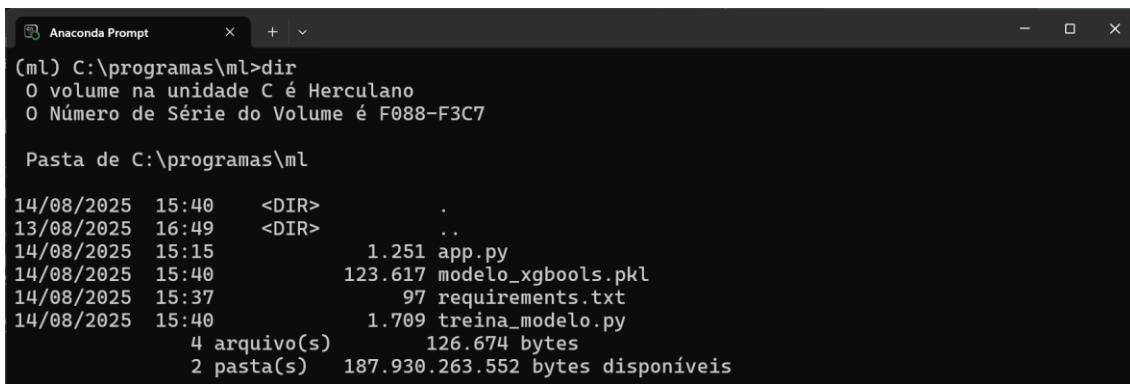
# Grava em disco o modelo treinado em um arquivo no formato 'pickle'
joblib.dump(modelo, 'modelo_xgboost.pkl')
```

Salve e execute, o resultado deverá ser semelhante ao abaixo:



```
(ml) C:\programas\ml>python treina_modelo.py
A acurácia do modelo é de 83.61%
```

Pode-se ver o modelo salvo no diretório:



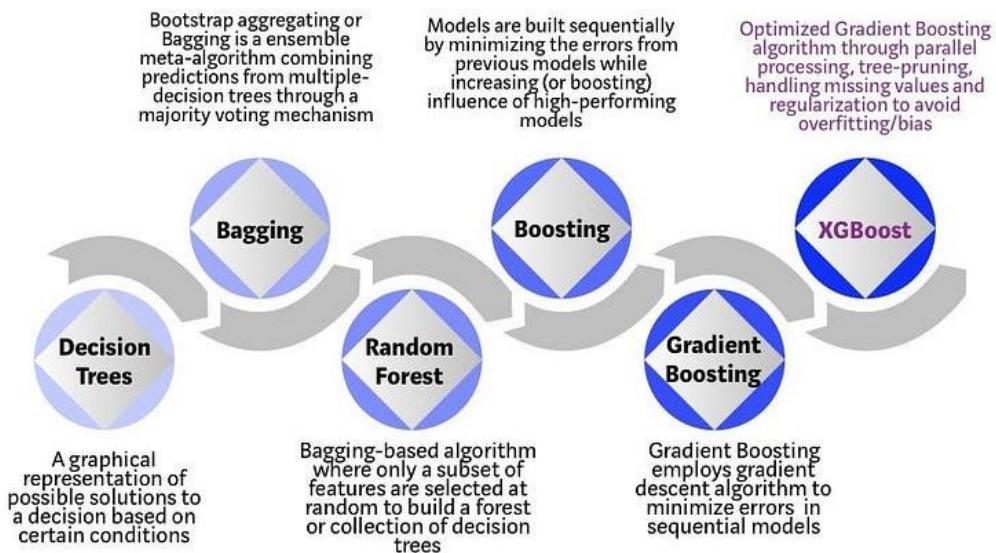
```
(ml) C:\programas\ml>dir
O volume na unidade C é Herculano
O Número de Série do Volume é F088-F3C7

Pasta de C:\programas\ml

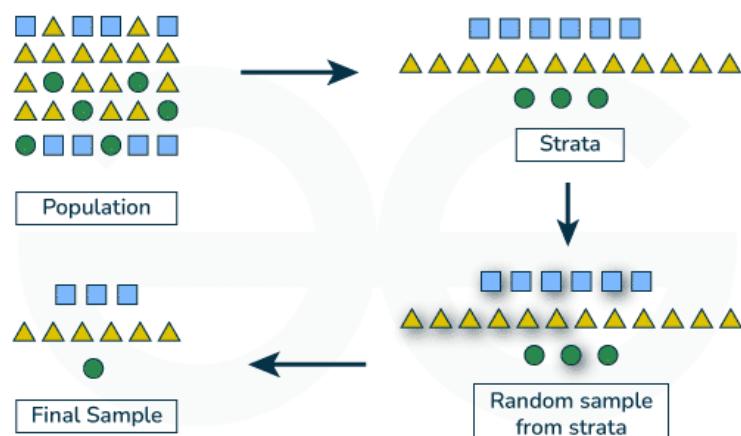
14/08/2025 15:40    <DIR>      .
13/08/2025 16:49    <DIR>      ..
14/08/2025 15:15            1.251 app.py
14/08/2025 15:40            123.617 modelo_xgboost.pkl
14/08/2025 15:37            97 requirements.txt
14/08/2025 15:40            1.709 treina_modelo.py
        4 arquivo(s)       126.674 bytes
        2 pasta(s)     187.930.263.552 bytes disponíveis
```

Explicações:

- [XGBoost \(eXtreme Gradient Boosting\)](#): Árvores de decisão com reforço gradativo usando gradiente descendente.



- **stratify**: A amostragem **estratificada** é uma técnica de amostragem na qual a população é subdividida em grupos com base em características específicas relevantes para o problema antes da amostragem. As amostras são retiradas desse grupo com tamanhos amplos proporcionais ao tamanho do subgrupo na população e combinadas para formar a amostra final. O objetivo é garantir que todos os subgrupos sejam representados proporcionalmente na amostra final.



11 Criação do Formulário

11.1 Aplicativo Python

Tela de entrada de dados:

O código abaixo cria o formulário de entrada de dados (arquivo *app_formulario.py*):

```
# -*- coding: utf-8 -*-
from dash import Dash, dcc, html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc

app = Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

formulario = dbc.Container([
    dbc.Row([
        dbc.Col([
            dbc.CardGroup([
                dbc.Label('Idade'),
                dbc.Input(id='idade', type='number',
                          placeholder='Digite a idade')
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Sexo biológico'),
                dbc.Select(id='sexo', options=[
                    {'label': 'Masculino', 'value': '1'},
                    {'label': 'Feminino', 'value': '0'}
                ])
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Tipo de dor no peito'),
                dbc.Select(id='cp', options=[
                    {'label': 'Angina típica', 'value': '1'},
                    {'label': 'Angina atípica', 'value': '2'},
                    {'label': 'Dor não cardíaca', 'value': '3'},
                    {'label': 'Assintomático', 'value': '4'}
                ])
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Pressão arterial em repouso'),
                dbc.Input(id='trestbps', type='number',
                          placeholder='Digite a pressão arterial em repouso')
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Colesterol sérico'),
                dbc.Input(id='chol', type='number',
                          placeholder='Digite o colesterol sérico')
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Glicemia em jejum'),
                dbc.Select(id='fbs', options=[
                    {'label': 'Nenhum', 'value': '0'},
                    {'label': 'Muito baixa', 'value': '1'},
                    {'label': 'Baixa', 'value': '2'},
                    {'label': 'Normal', 'value': '3'},
                    {'label': 'Alta', 'value': '4'},
                    {'label': 'Muito alta', 'value': '5'}
                ])
            ], className='mb-3')
        ],
        dbc.Col([
            dbc.Label('Frequência cardíaca máxima atingida'),
            dbc.Input(id='age', type='number',
                      placeholder='Digite a frequência cardíaca máxima')
        ],
        style={'flex-grow': 1})
    ],
    style={'display': 'flex', 'flex-wrap': 'wrap'}
)])
```

```

        {'label': 'Menor que 120 mg/dl', 'value': '0'},
        {'label': 'Maior que 120 mg/dl', 'value': '1'},
    ])
], className='mb-3'),
dbc.CardGroup([
    dbc.Label('Eletrocardiografia em repouso'),
    dbc.Select(id='restecg', options=[
        {'label': 'Normal', 'value': '0'},
        {'label': 'Anormalidades de ST-T', 'value': '1'},
        {'label': 'Hipertrofia ventricular esquerda', 'value': '2'},
    ])
], className='mb-3'),
]),
dbc.Col([
    dbc.CardGroup([
        dbc.Label('Frequência cardíaca máxima atingida'),
        dbc.Input(id='thalach', type='number',
            placeholder='Digite a frequência cardíaca máxima atingida')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Angina induzida por exercício'),
        dbc.Select(id='exang', options=[
            {'label': 'Não', 'value': '0'},
            {'label': 'Sim', 'value': '1'},
        ])
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Depressão do segmento ST induzida por exercício'),
        dbc.Input(id='oldpeak', type='number',
            placeholder='Depressão do segmento ST induzida por exercício')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Depressão do segmento ST'),
        dbc.Select(id='slope', options=[
            {'label': 'Ascendente', 'value': '1'},
            {'label': 'Plana', 'value': '2'},
            {'label': 'Descendente', 'value': '3'},
        ])
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Número de vasos principais coloridos por fluoroscopia'),
        dbc.Select(id='ca', options=[
            {'label': '0', 'value': '0'},
            {'label': '1', 'value': '1'},
            {'label': '2', 'value': '2'},
            {'label': '3', 'value': '3'},
        ])
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Cintilografia do miocárdio'),
        dbc.Select(id='thal', options=[
            {'label': 'Normal', 'value': '3'},
            {'label': 'Defeito fixo', 'value': '6'},
            {'label': 'Defeito reversível', 'value': '7'},
        ])
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Button('Prever', id='botao-prever', color='primary')
    ], className='mb-3')
])
])
])

app.layout = html.Div([
    html.H1('Previsão de doença cardíaca'),
    formulario
])
)

app.run(debug=True)

```

Componentes:

- `dbc.Container()`: Elemento que contém outros.
- `dbc.Row() / Col()`: Cria uma linha / coluna dentro do contêiner.
- `dbc.CardGroup()`: Agrupa componentes adicionando um espaço.
- `layout`: *Layout da aplicação*.
- `Label() / Input() / Button() / Select()`: Análogos Dash dos componentes de formulário do HTML.

11.2 Componentes Dash Bootstrap

A biblioteca [Dash Bootstrap Components](#) une as facilidades da popular biblioteca [Bootstrap](#), um *framework* de *front-end* para desenvolvimento *web* que oferece componentes estilizados e responsivos, com as funcionalidades da Dash.

Ao integrar o Bootstrap à Dash, é possível criar *interfaces* de usuário elegantes e adaptáveis com menos esforço, aproveitando a ampla gama de componentes prontos para uso do Bootstrap e a interatividade e facilidade de uso da Dash. Isso torna o desenvolvimento *web* mais eficiente e acessível para quem já trabalha com Python.

Para explorar mais a fundo as capacidades e os recursos da biblioteca Dash Bootstrap Components, considere visitar os seguintes *links* abaixo, que oferecem uma ampla gama de informações:

- [Documentação da biblioteca Dash Bootstrap Components](#): Documentação principal da biblioteca Dash Bootstrap Components, onde você pode encontrar informações detalhadas sobre como começar a usar a biblioteca em seus projetos Dash. A documentação abrange desde a instalação e configuração inicial até exemplos avançados de uso.
- [Temas disponíveis](#): Lista de temas disponíveis que podem ser utilizados com a Dash Bootstrap Components. Esta seção é particularmente útil para personalizar a aparência de suas aplicações Dash, permitindo escolher entre uma variedade de temas pré-definidos baseados no Bootstrap. Cada tema vem com sua própria paleta de cores e estilos, facilitando a harmonização visual de sua aplicação. Confira os [temas disponíveis](#).
- [Componentes](#): Seção de componentes da documentação, onde é possível encontrar uma descrição detalhada de todos os componentes disponíveis. Inclui informações sobre a funcionalidade de cada componente, propriedades configuráveis e exemplos de código para sua implementação. Se você está procurando por componentes específicos, como botões, cartões, formulários, ou qualquer outro elemento para enriquecer suas aplicações Dash, este é o lugar para explorar.

Alguns dos componentes disponíveis são:

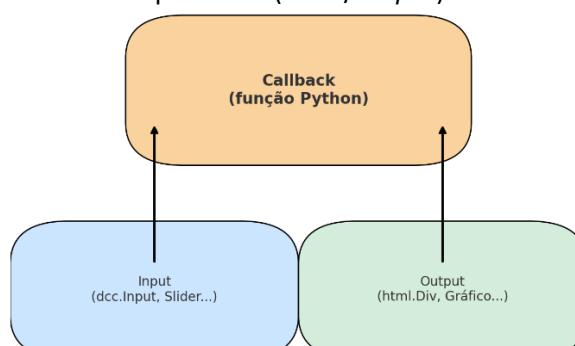
- | | | | |
|---------------|----------------|---------------|------------|
| ○ Accordion | ○ Collapse | ○ ListGroup | ○ Progress |
| ○ Alert | ○ DropdownMenu | ○ Modal | ○ Spinner |
| ○ Badge | ○ Fade | ○ Nav | ○ Table |
| ○ Breadcrumb | ○ Form | ○ Navbar | ○ Tabs |
| ○ Button | ○ Input | ○ Offcanvas | ○ Toast |
| ○ ButtonGroup | ○ InputGroup | ○ Pagination | ○ Tooltip |
| ○ Card | ○ Jumbotron | ○ Placeholder | |
| ○ Carousel | ○ Layout | ○ Popover | |

11.3 Callbacks

Outro conceito importante é o de [callback](#), que são funções que são automaticamente acionadas em resposta a alguma ação do usuário, como uma mudança de valor em um componente do *dashboard* ou o clique em um botão. Um *callback* no Dash conecta eventos de *interface*, como entradas (*inputs*) a atualizações automáticas em outros componentes (saídas desejadas – *outputs*), tudo via Python, sem precisar de JavaScript, permitindo a atualização dinâmica dos componentes da interface, como gráficos e tabelas, sem a necessidade de recarregar a página.

Em resumo, um *callback* é basicamente uma função Python decorada que:

- Escuta mudanças em um ou mais componentes da interface (entrada/*input*)
- Atualiza automaticamente outros componentes (saída/*output*)



12 Exibindo as Previsões do Modelo

Início do arquivo *app_formulario.py*:

```
# -*- coding: utf-8 -*-
from dash import Dash, dcc, html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
import joblib
import pandas as pd
import numpy as np

# Carrega o modelo de IA
modelo = joblib.load('modelo_xgboost.pkl')
```

Final do arquivo:

```
        dbc.CardGroup([
            dbc.Button('Prever', id='botao-prever', color='primary', n_clicks=0)
        ], className='mb-3')
    ])
])

app.layout = html.Div([
    html.H1('Previsão de doença cardíaca'),
    formulario,
    html.Div(id='previsao')
])

@app.callback(
    Output('previsao', 'children'),
    [Input('botao-prever', 'n_clicks')], # Chama a função sempre que for clicado no botão 'Prever'
    [State('idade', 'value'),
     State('sexo', 'value'),
     State('cp', 'value'),
     State('trestbps', 'value'),
     State('chol', 'value'),
     State('fbs', 'value'),
     State('restecg', 'value'),
     State('thalach', 'value'),
     State('exang', 'value'),
     State('oldpeak', 'value'),
     State('slope', 'value'),
     State('ca', 'value'),
     State('thal', 'value')]
)
def prever_doenca(n_clicks, idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal):
    if n_clicks == 0:
        return ''

    # Cria um DataFrame e já preenche seus dados com os valores do formulário
    entradas_usuario = pd.DataFrame(
        data = [[idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]],
        columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
    )

    # oldpeak é float
    entradas_usuario['oldpeak'] = entradas_usuario['oldpeak'].astype(np.float64)

    # Converte números em formato string para int (exceto a coluna oldpeak tratada acima)
    for col in entradas_usuario.columns:
        if col != 'oldpeak':
            entradas_usuario[col] = entradas_usuario[col].astype(int)

    previsao = modelo.predict(entradas_usuario)[0]
    if previsao == 1:
        return html.H2('Você tem doença cardíaca!')
    else:
        return html.H2('Você não tem doença cardíaca!')

app.run(debug=True)
```

Explicações:

- `n_clicks`: Utilizado para controle no callback.
- `astype()`: Transforma o tipo do dado.
- `mb_3`: Margem inferior do componente.
- `@app.callback()`: Esta decoração especifica que a função é chamada quando o botão é clicado (`Input('botao-prever', 'n_clicks')`) e utiliza o valor atual nos campos de entrada como um estado (`campos state()`). `State()` permite passar valores extras como parâmetros sem disparar as callbacks. Dessa forma pode-se receber os dados do formulário somente quando se clicar no botão e não quando cada campo for modificado.
- `modelo.predict(entradas_usuario) [0]`: Faz a predição usando a primeira linha de dados do DataFrame (ou seja, os dados do formulário).

Com isso, se todos os campos forem preenchidos, o sistema já fornecerá uma resposta:

- Exemplo ‘com’ doença cardíaca:

Previsão de doença cardíaca

Idade: 70 | Frequência cardíaca máxima atingida: 200

Sexo biológico: Masculino | Angina induzida por exercício: Não

Tipo de dor no peito: Angina típica | Depressão do segmento ST induzida por exercício: 2

Pressão arterial em repouso: 100 | Depressão do segmento ST: Ascendente

Colesterol sérico: 200 | Número de vasos principais coloridos por fluoroscopia: 0

Glicemia em jejum: Menor que 120 mg/dl | Cintilografia do miocárdio: Normal

Eletrocardiografia em repouso: Anormalidades de ST-T | Prever

Você não tem doença cardíaca!

- Exemplo ‘sem’ doença cardíaca:

Previsão de doença cardíaca

Idade: 70 | Frequência cardíaca máxima atingida: 200

Sexo biológico: Masculino | Angina induzida por exercício: Sim

Tipo de dor no peito: Angina típica | Depressão do segmento ST induzida por exercício: 2

Pressão arterial em repouso: 100 | Depressão do segmento ST: Ascendente

Colesterol sérico: 200 | Número de vasos principais coloridos por fluoroscopia: 1

Glicemia em jejum: Maior que 120 mg/dl | Cintilografia do miocárdio: Normal

Eletrocardiografia em repouso: Anormalidades de ST-T | Prever

Você tem doença cardíaca!

Mas, se algum campo não for preenchido, o erro abaixo ocorrerá.

Você tem doença cardíaca!

Errors

Callback error updating previsao.children

TypeError: int() argument must be a string, a bytes-like object or a real number, not 'NoneType'

Traceback (most recent call last)

File "C:\programas\iml\app_formulario.py", line 162, in prever_doenca
entradas_usuario[col] = entradas_usuario[col].astype(int)
~~~~~  
File "C:\Users\debiashanacarol\envs\iml\site-packages\pandas\core\generic.py", line 6643, in astype  
new\_data = self.\_mgr.astype(dtype=dtype, copy=copy,  
errors=errors)  
~~~~~  
File "C:\Users\debiashanacarol\envs\iml\site-packages\pandas\core\internal\managers.py", line 430, in astype
return self.apply()
~~~~~  
File "C:\Users\debiashanacarol\envs\iml\site-packages\pandas\core\internal\managers.py", line 363, in apply  
applied = getattr(b, f)(\*kargs)  
~~~~~

Isso ocorre por que o sistema está esperando algum valor (*string*) mas nenhum (*None*) está sendo fornecido.

Essa é uma situação comum em sistemas de *machine learning*, informações incompletas. A solução adotada neste projeto será preencher os valores faltantes usando as medianas dos valores calculados no modelo de treinamento. Isso será feito gerando um novo arquivo *pkl* no arquivo *treina_modelo.py*:

Final do arquivo *treina_modelo.py*:

```
# Biblioteca com várias funções úteis para processamento paralelo
import joblib

# Grava em disco o modelo treinado em um arquivo no formato 'pickle'
medianas = X.median() # Calcula as medianas
joblib.dump(modelo, 'modelo_xgboost.pkl') # Grava em disco o modelo treinado
joblib.dump(medianas, 'medianas.pkl') # Grava em disco o modelo com as medianas calculadas
```

Arquivo *app_formulario.py*:

```
# Carrega o modelo de IA
modelo = joblib.load('modelo_xgboost.pkl')
medianas = joblib.load('medianas.pkl')

.

.

.

def prever_doenca(n_clicks, idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal):
    if n_clicks == 0:
        return ''

    entradas_usuario = pd.DataFrame(
        data = [[idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]],
        columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal'])

    # Preenche valores em branco com as medianas
    entradas_usuario.fillna(medianas, inplace=True)

    # oldpeak é float
    entradas_usuario['oldpeak'] = entradas_usuario['oldpeak'].astype(np.float64)

.
```

Explicações:

- `fillna()`: Preenche os dados faltantes com as medianas calculadas anteriormente

Agora, mesmo fornecendo poucos valores, o sistema irá fornecer um resultado:

- Exemplo ‘com’ doença cardíaca:

Previsão de doença cardíaca

Idade: 70

Frequência cardíaca máxima atingida: Digite a frequência cardíaca máxima atingida

Sexo biológico: Masculino

Angina induzida por exercício: Depressão do segmento ST induzida por exercício

Tipo de dor no peito: Unstable angina

Depressão do segmento ST induzida por exercício: Depressão do segmento ST

Pressão arterial em repouso: 140 mmHg

Número de vasos principais coloridos por fluoroscopia: 0

Colesterol sérico: 230 mg/dL

Cintilografia do miocárdio: Normal

Glicemia em jejum: Menor que 120 mg/dL

Eletrocardiografia em repouso: Normal

Prever

Você não tem doença cardíaca!

Errors | Callbacks | v3.2.0 | Server | (refresh)

- Exemplo ‘sem’ doença cardíaca:

Previsão de doença cardíaca

Idade: 70

Frequência cardíaca máxima atingida: Digite a frequência cardíaca máxima atingida

Sexo biológico: Masculino

Angina induzida por exercício: Depressão do segmento ST induzida por exercício

Tipo de dor no peito: Unstable angina

Depressão do segmento ST induzida por exercício: Depressão do segmento ST

Pressão arterial em repouso: 140 mmHg

Número de vasos principais coloridos por fluoroscopia: 1

Colesterol sérico: 230 mg/dL

Cintilografia do miocárdio: Normal

Glicemia em jejum: Menor que 120 mg/dL

Eletrocardiografia em repouso: Normal

Prever

Você tem doença cardíaca!

Errors | Callbacks | v3.2.0 | Server | (refresh)

13 Melhorando a Apresentação do Formulário

Dentro da pasta do projeto crie outra chamada `assets` e dentro dela crie o arquivo `style.css` com o conteúdo abaixo, que personalizada estilos CSS:

```
body {                                     /* Cor do background geral do dashboard */
    background-color: #cfcd5db;           /* Cinza claro */
    font-family: 'Segoe UI', sans-serif; /* Fonte Segoe UI */
}

.custom-title {      /* Formatação do título */
    color: #153d58;        /* Azul */
    font-weight: bold;     /* Negrito */
}

.custom-subtitle {   /* Formatação do subtítulo */
    color: #343a40;        /* Cinza */
    font-style: italic;    /* Itálico */
}

.custom-label {     /* Formatação do label */
}

.custom-select {    /* Cor do select e alterações de estilo */
    color: blue;
    font-weight: bold;
    background-color: transparent; /* Fundo transparente */
}

.custom-select option { /* Cor das opções do select */
    background-color: #cfcd2d6; /* Cinza claro para o fundo */
    color: #343a40;           /* Cinza escuro para a fonte */
}

.custom-input {      /* Cor do input e alterações de estilo */
    color: blue;
    font-weight: bold;
    background-color: transparent;
}

::placeholder {      /* Cor do placeholder do input e alterações de estilo */
    font-style: italic;
    font-weight: normal;
    background-color: transparent;
}
```

O código abaixo altera o tema para `FLATLY` e aplica classes CSS nos elementos. Além disso muda a mensagem de previsão para um componente Alert.

```
# -*- coding: utf-8 -*-
from dash import Dash, html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
import joblib
import pandas as pd
import numpy as np

# Carrega o modelo de IA
modelo = joblib.load('modelo_xgboost.pkl')
medianas = joblib.load('medianas.pkl')

app = Dash(__name__,
           external_stylesheets=[dbc.themes.FLATLY])

formulario = dbc.Container([
    html.P("Preencha as informações abaixo e clique no botão 'Prever' para executar o modelo",
           className='text-center mb-4 custom-subtitle'),
    dbc.Row([
        dbc.Col([
            dbc.CardGroup([
                dbc.Label('Idade', className='custom-label'),
                dbc.Input(id='idade', type='number',
                          placeholder='Digite a idade',
                          className='custom-input')
            ], className='mb-3'),
            dbc.CardGroup([
                dbc.Label('Sexo biológico', className='custom-label'),

```

```

        dbc.Select(id='sexo', options=[
            {'label': 'Masculino', 'value': '1'},
            {'label': 'Feminino', 'value': '0'}
        ], class_name='custom-select')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Tipo de dor no peito', className='custom-label'),
        dbc.Select(id='cp', options=[
            {'label': 'Angina típica', 'value': '1'},
            {'label': 'Angina atípica', 'value': '2'},
            {'label': 'Dor não cardíaca', 'value': '3'},
            {'label': 'Assintomático', 'value': '4'}
        ], class_name='custom-select')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Pressão arterial em repouso', className='custom-label'),
        dbc.Input(id='trestbps', type='number',
            placeholder='Digite a pressão arterial em repouso',
            class_name='custom-input')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Colesterol sérico', className='custom-label'),
        dbc.Input(id='chol', type='number',
            placeholder='Digite o colesterol sérico',
            class_name='custom-input')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Glicemia em jejum', className='custom-label'),
        dbc.Select(id='fbs', options=[
            {'label': 'Menor que 120 mg/dl', 'value': '0'},
            {'label': 'Maior que 120 mg/dl', 'value': '1'}
        ], class_name='custom-select')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Eletrocardiografia em repouso', className='custom-label'),
        dbc.Select(id='restecg', options=[
            {'label': 'Normal', 'value': '0'},
            {'label': 'Anormalidades de ST-T', 'value': '1'},
            {'label': 'Hipertrofia ventricular esquerda', 'value': '2'}
        ], class_name='custom-select')
    ], className='mb-3'),
]),
dbc.Col([
    dbc.CardGroup([
        dbc.Label('Frequência cardíaca máxima atingida', className='custom-label'),
        dbc.Input(id='thalach', type='number',
            placeholder='Digite a frequência cardíaca máxima atingida',
            class_name='custom-input')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Angina induzida por exercício', className='custom-label'),
        dbc.Select(id='exang', options=[
            {'label': 'Não', 'value': '0'},
            {'label': 'Sim', 'value': '1'}
        ], class_name='custom-select')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Depressão do segmento ST induzida por exercício', className='custom-label'),
        dbc.Input(id='oldpeak', type='number',
            placeholder='Depressão do segmento ST induzida por exercício',
            class_name='custom-input')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Depressão do segmento ST', className='custom-label'),
        dbc.Select(id='slope', options=[
            {'label': 'Ascendente', 'value': '1'},
            {'label': 'Plana', 'value': '2'},
            {'label': 'Descendente', 'value': '3'}
        ], class_name='custom-select')
    ], className='mb-3'),
    dbc.CardGroup([
        dbc.Label('Número de vasos principais coloridos por fluoroscopia', className='custom-label'),
        dbc.Select(id='ca', options=[
            {'label': '0', 'value': '0'},
            {'label': '1', 'value': '1'},
            {'label': '2', 'value': '2'},
            {'label': '3', 'value': '3'}
        ], class_name='custom-select')
    ])
])

```

```

        ], class_name='custom-select')
    ], className='mb-3'),
dbc.CardGroup([
    dbc.Label('Cintilografia do miocárdio', className='custom-label'),
    dbc.Select(id='thal', options=[
        {'label': 'Normal', 'value': '3'},
        {'label': 'Defeito fixo', 'value': '6'},
        {'label': 'Defeito reversível', 'value': '7'},
    ], class_name='custom-select')
], className='mb-3'),
dbc.Button('Prever', id='botao-prever', color='success', n_clicks=0,
            className='mb-3 mt-3')
])
])
],
fluid=True)

app.layout = html.Div([
    html.H1('Previsão de Doença Cardíaca', className='text-center mt-4 custom-title'),
    formulario,
    html.Div(id='previsao')
])

@app.callback(
    Output('previsao', 'children'),
    [Input('botao-prever', 'n_clicks')],
    [State('idade', 'value'),
     State('sexo', 'value'),
     State('cp', 'value'),
     State('trestbps', 'value'),
     State('chol', 'value'),
     State('fbs', 'value'),
     State('restecg', 'value'),
     State('thalach', 'value'),
     State('exang', 'value'),
     State('oldpeak', 'value'),
     State('slope', 'value'),
     State('ca', 'value'),
     State('thal', 'value')])
)
def prever_doenca(n_clicks, idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak,
slope, ca, thal):
    if n_clicks == 0:
        return ''

    entradas_usuario = pd.DataFrame(
        data = [[idade, sexo, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal]],
        columns = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang',
'oldpeak', 'slope', 'ca', 'thal']
    )

    # Preenche valores em branco com as medianas
    entradas_usuario.fillna(medianas, inplace=True)
    #entradas_usuario.infer_objects(copy=False)

    # oldpeak é float
    entradas_usuario['oldpeak'] = entradas_usuario['oldpeak'].astype(np.float64)

    # Converte números em formato string para int
    for col in entradas_usuario.columns:
        if col != 'oldpeak':
            entradas_usuario[col] = entradas_usuario[col].astype(int)

    previsao = modelo.predict(entradas_usuario)[0]
    if previsao == 1:
        mensagem = 'Você tem doença cardíaca!'
        cor_alerta = 'danger'
    else:
        mensagem = 'Você não tem doença cardíaca!'
        cor_alerta = 'secondary'

    alerta = dbc.Alert(mensagem, color=cor_alerta, className='d-flex justify-content-center mb-5')
    return alerta

app.run(debug=True)

```

Explicações:

- FLATLY: Tema com *design* limpo e plano com foco na legibilidade e uma paleta de cores suaves.
- Alert(): Componente que mostra mensagens de *feedback* na tela.
- fluid: Contêiner de largura total que abrange toda a largura da janela de visualização e redimensiona fluidamente com a janela do navegador. Isso remove as margens padrão e o preenchimento aplicados pelo contêiner de largura fixa.

O formulário deverá estar agora semelhante à versão abaixo:

- Exemplo ‘com’ doença cardíaca:

Previsão de Doença Cardíaca
Preencha as informações abaixo e clique no botão 'Prever' para executar o modelo

Idade 70	Frequência cardíaca máxima atingida <i>Digite a frequência cardíaca máxima atingida</i>
Sexo biológico Masculino	Angina induzida por exercício <i>Depressão do segmento ST induzida por exercício</i>
Tipo de dor no peito <i>Depressão do segmento ST</i>	Depressão do segmento ST induzida por exercício <i>Depressão do segmento ST induzida por exercício</i>
Pressão arterial em repouso <i>Digite a pressão arterial em repouso</i>	Número de vasos principais coloridos por fluoroscopia 2
Colesterol sérico <i>Digite o colesterol sérico</i>	Cintilografia do miocárdio <i>Cintilografia do miocárdio</i>
Glicemia em jejum <i>Cintilografia do miocárdio</i>	Eletrocardiografia em repouso <i>Digitate a eletrocardiografia em repouso</i>

Prever

Você tem doença cardíaca!

- Exemplo ‘sem’ doença cardíaca:

Previsão de Doença Cardíaca
Preencha as informações abaixo e clique no botão 'Prever' para executar o modelo

Idade 70	Frequência cardíaca máxima atingida <i>Digite a frequência cardíaca máxima atingida</i>
Sexo biológico Masculino	Angina induzida por exercício <i>Depressão do segmento ST induzida por exercício</i>
Tipo de dor no peito <i>Depressão do segmento ST</i>	Depressão do segmento ST induzida por exercício <i>Depressão do segmento ST induzida por exercício</i>
Pressão arterial em repouso <i>Digite a pressão arterial em repouso</i>	Número de vasos principais coloridos por fluoroscopia <i>Cintilografia do miocárdio</i>
Colesterol sérico <i>Digite o colesterol sérico</i>	Cintilografia do miocárdio <i>Cintilografia do miocárdio</i>
Glicemia em jejum <i>Cintilografia do miocárdio</i>	Eletrocardiografia em repouso <i>Digitate a eletrocardiografia em repouso</i>

Prever

Você não tem doença cardíaca!

14 Criação da Página Principal

Crie o arquivo `main.py` com o conteúdo abaixo, que utilizará recursos de [barra de navegação](#):

```
# -*- coding: utf-8 -*-
from dash import Dash, dcc, html
from dash.dependencies import Input, Output
import dash_bootstrap_components as dbc

app = Dash(__name__, external_stylesheets=[dbc.themes.FLATLY, './assets/style.css'])

navegacao = dbc.NavbarSimple(
    children=[
        dbc.NavItem(dbc.NavLink('Gráficos', href='/graficos')),
        dbc.NavItem(dbc.NavLink('Formulário', href='/formulario')),
    ],
    brand='Unoesc',
    brand_href='/',
    color='primary',
    dark=True
)

app.layout = html.Div([
    dcc.Location(id='url', refresh=False),
    navegacao,
    html.Div(id='conteudo')
])

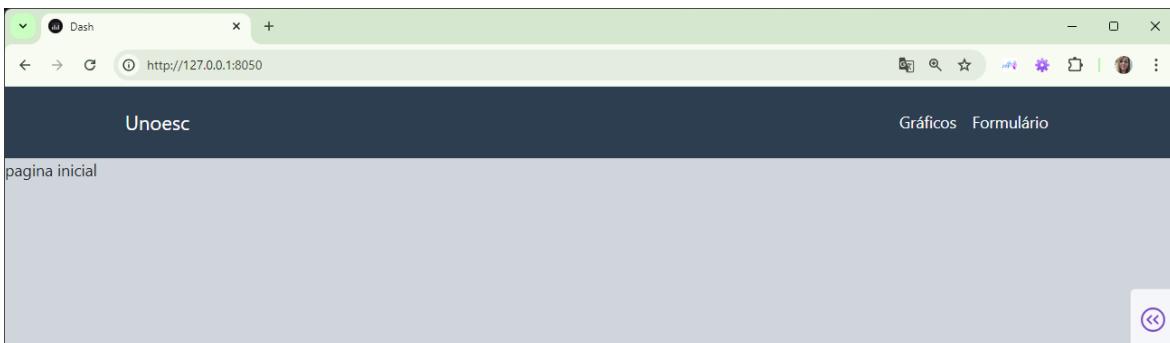
@app.callback(
    Output('conteudo', 'children'),
    [Input('url', 'pathname')]
)
def mostrar_pagina(pathname):
    if pathname == '/formulario':
        return html.P('formulario')
    elif pathname == '/graficos':
        return html.P('graficos')
    else:
        return html.P('pagina inicial')

app.run(debug=True)
```

No *prompt* do Anaconda, interrompa a execução anterior e execute o seguinte:

```
➤ python main.py
```

A página deverá estar agora semelhante à versão abaixo:



Explicações:

- `'./assets/style.css'`: Informa local e nome do arquivo .CSS
- `dbc.NavbarSimple()`: Elemento de barra de navegação superior do Bootstrap
- `dbc.NavItem()`: Item de uma Navbar
- `dbc.NavLink()`: Link em um item de uma Navbar
- `brand='Unoesc'`: Logotipo/marca na barra de navegação
- `brand_href='/'`: Link para quando for clicado na marca/logotipo
- `pathname`: Representa o ‘caminho’ (URL) da página, como `/graficos` ou `/formulario`
- `dcc.Location(id='url', refresh=False)`: Usado para integrar a navegação baseada em URL na aplicação Dash sem recarregar a página web. A opção `refresh=False` significa que as mudanças no URL não causarão o recarregamento da página inteira. Isso é fundamental para criar uma experiência de usuário fluida e eficiente em aplicações.

15 Integração da Página Inicial com a de Gráficos e de Formulário

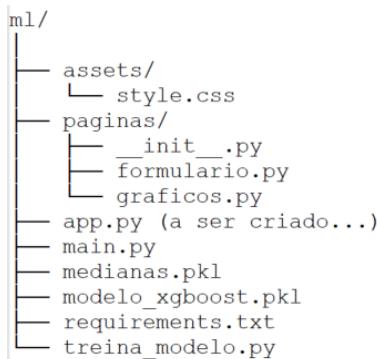
15.1 Estrutura do Projeto

Dentro da pasta do projeto crie outra chamada *paginas* e mova os arquivos *app_graficos.py* e *app_formulario.py* para dentro dela. Renomeie os arquivos para *graficos.py* e *formulario.py* respectivamente.

Dentro da pasta *paginas* crie o arquivo *_init_.py* (atenção, são dois sublinhados antes e depois de *init*):

```
# -*- coding: utf-8 -*-
from . import graficos
from . import formulario
```

Estrutura do projeto:



Descrição dos componentes:

- Pasta *paginas*: Contém módulos Python responsáveis pela lógica de exibição das diferentes seções do dashboard. O arquivo *_init_.py* indica que esta pasta é um pacote Python, permitindo a importação de seus módulos (*formulario.py* e *graficos.py*) de forma organizada.
- Pasta *assets*: Armazena arquivos estáticos, como CSS, que são usados para estilizar o *dashboard*.
- *formulario.py*: Contém o formulário de entrada de dados para previsões de *machine learning*.
- *graficos.py*: Responsável por gerar visualizações gráficas dos dados.
- *app.py*: Monta toda a estrutura do *dashboard*, incluindo visualizações e formulários, importando módulos de *paginas*.
- *requirements.txt*: Lista todas as bibliotecas Python necessárias para executar o projeto.
- *modelo_xgboost.pkl* e *medianas.pkl*: Arquivos contendo o modelo de machine learning treinado e outros dados necessários para as previsões.
- *treina_modelo.py*: Script para treinar o modelo de *machine learning* utilizado no projeto.

15.2 Módulos e Pacotes no Python

É importante entender dois conceitos fundamentais em Python: módulos e pacotes.

- **Módulo:** Um módulo no contexto do Python é basicamente um arquivo contendo código Python. Este arquivo inclui tanto definições (como classes e funções) quanto instruções executáveis. O nome do módulo deriva diretamente do nome do arquivo, excluindo-se a extensão *.py*. Por exemplo, se for criado um arquivo chamado *calculadora.py*, este arquivo é considerado um módulo chamado *calculadora* no universo Python. A principal vantagem de usar módulos é a capacidade de reutilizar código. Em vez de escrever o mesmo código várias vezes, você pode defini-lo uma vez em um módulo e importá-lo sempre que necessário.
- **Pacote:** Quando o número de módulos começa a crescer, surge a necessidade de uma organização melhor. É aí que entram os pacotes. Um pacote é basicamente um diretório que serve para agrupar módulos relacionados. Para que um diretório seja reconhecido como um pacote pelo Python, ele deve conter um arquivo especial chamado *_init_.py*. Este arquivo pode estar vazio, mas sua presença é que sinaliza ao Python que o diretório deve ser tratado como um pacote. Isso permite organizar o código em uma estrutura hierárquica mais limpa e lógica, facilitando a manutenção e a compreensão do código. Por exemplo, um pacote de nome *utils* pode conter módulos como *math_utils.py* e *string_utils.py*, organizando funcionalidades relacionadas de forma intuitiva.

15.3 Adaptação da Página de Gráficos

Altere o código da página *graficos.py* para o abaixo:

```
# -*- coding: utf-8 -*-
from ucimlrepo import fetch_ucirepo
import plotly.express as px
from dash import dcc, html
import dash_bootstrap_components as dbc

# Importa os dados
heart_disease = fetch_ucirepo(id=45)
dados = heart_disease.data.features

# Acrescentando a coluna 'doença'
dados['doença'] = 1 * (heart_disease.data.targets > 0)

# Cria o histograma
fig_hist = px.histogram(dados, x='age', nbins=30, title='Distribuição Etária (Plotly)', color='doença')
fig_hist.update_layout(xaxis_title='Idade', yaxis_title='Frequência')

# Cria o boxplot
fig_box = px.box(dados, x='doença', y='age', title='Distribuição Etária por Doença', color='doença')
fig_box.update_layout(xaxis_title='Diagnóstico (0 = não | 1 = sim)', yaxis_title='Idade')

# Cria as divs
div_hist = html.Div([
    dcc.Graph(figure=fig_hist)
])

div_box = html.Div([
    dcc.Graph(figure=fig_box)
])

layout = html.Div([
    html.H1('Análise de Dados do UCI Repository Heart Disease',
           className='text-center mb-4 custom-subtitle'),
    dbc.Container([
        dbc.Row([
            dbc.Col([div_hist], md=7),
            dbc.Col([div_box], md=5)
        ])
    ])
])
```

Esse código removeu as seguintes linhas:

```
app = Dash(__name__)
app.run(debug=True)
```

Além disso, a linha

```
app.layout = html.Div([
```

Foi alterada para

```
layout = html.Div([
```

Com essas modificações, esse arquivo de código-fonte Python não funciona mais como uma aplicação separada, mas sim como um módulo da aplicação principal.

Explicações:

```
dbc.Container([
    dbc.Row([
        dbc.Col([div_hist], md=7),
        dbc.Col([div_box], md=5)
    ])
])
```

O trecho acima altera a visualização de forma a mostrar os dois gráficos na mesma linha.

Os parâmetros `md=7` e `md=5` servem para indicar a proporção de tela de cada elemento. A soma precisa totalizar 12 e isto é devido à forma de funcionamento da [grade \(grid\) do Bootstrap](#).

15.4 Adaptação da Página de Formulário

A página de formulário precisa criar um *callback* para o aplicativo. Mas, como registrar no aplicativo do Dash um *callback* se não se tem acesso à variável *app*? É precisamos de um acesso à variável *app* definida no programa principal. Quando se está usando o Dash com uma aplicação inicial, é preciso compartilhá-la com suas subpáginas ou com suas outras páginas. Por esse motivo, a integração irá exigir mudanças nos arquivos principais da aplicação.

O arquivo *app.py* terá somente a declaração da variável *app*. Já o *main.py* importará o *app* (`from app import app`) e além disso conterá a página principal, que tem o *menu* e todo o resto.

Crie o arquivo *app.py* e insira nele o seguinte:

```
# -*- coding: utf-8 -*-
from dash import Dash
import dash_bootstrap_components as dbc

app = Dash(__name__,
           suppress_callback_exceptions=True,
           external_stylesheets=[dbc.themes.FLATLY, './assets/style.css'])
```

O código acrescenta o parâmetro `suppress_callback_exceptions` igual a `True`. Caso contrário o navegador iria mostrar várias mensagens de erro, tanto na página de gráficos quanto na de formulários. Na página de gráficos por exemplo, o erro informaria que não existe o ID na página e não existe o botão ‘Prever’. Mas nem existe um botão prever na página de gráficos! Isso porque estamos pedindo a página principal, o *main*. O *main* carrega o módulo *app* e também carrega o módulo *paginas*. O módulo *paginas* carrega o formulário, que, por sua vez, tenta acessar o `@app`. Uma vez inicializado, o `@app` tenta procurar um `input` chamado `botao-prever` na tela e não acha. Afinal, estamos na tela de gráficos.

Para resolver esse problema, o parâmetro deve ser configurado no *app.py* para informar que quando carregar a *app*, na hora que definimos o Dash, deve-se ignorar *callbacks* que ainda não estiverem definidos. Dessa forma, ele não vai nos avisar se tiver uma *exception* desse tipo.

Altere o arquivo *main.py* para a versão abaixo:

```
# -*- coding: utf-8 -*-
from dash import dcc, html
from dash.dependencies import Input, Output
import dash_bootstrap_components as dbc
import paginas
from app import app

navegacao = dbc.NavbarSimple(
    children=[
        dbc.NavItem(dbc.NavLink('Gráficos', href='/graficos')),
        dbc.NavItem(dbc.NavLink('Formulário', href='/formulario')),
    ],
    brand='Dashboard - Unoesc',
    brand_href='/',
    color='primary',
    dark=True
)

app.layout = html.Div([
    dcc.Location(id='url', refresh=False),
    navegacao,
    html.Div(id='conteudo')
])

@app.callback(
    Output('conteudo', 'children'),
    [Input('url', 'pathname')])
def mostrar_pagina(pathname):
    if pathname == '/formulario':
        return paginas.formulario.layout
    elif pathname == '/graficos':
        return paginas.graficos.layout
    else:
        return html.P('pagina inicial')

app.run(debug=True)
```

Altere o código da página *formulario.py* para:

```
# -*- coding: utf-8 -*-
from dash import html
from dash.dependencies import Input, Output, State
import dash_bootstrap_components as dbc
import joblib
import pandas as pd
import numpy as np
from app import app
```

E:

```
layout = html.Div([
```

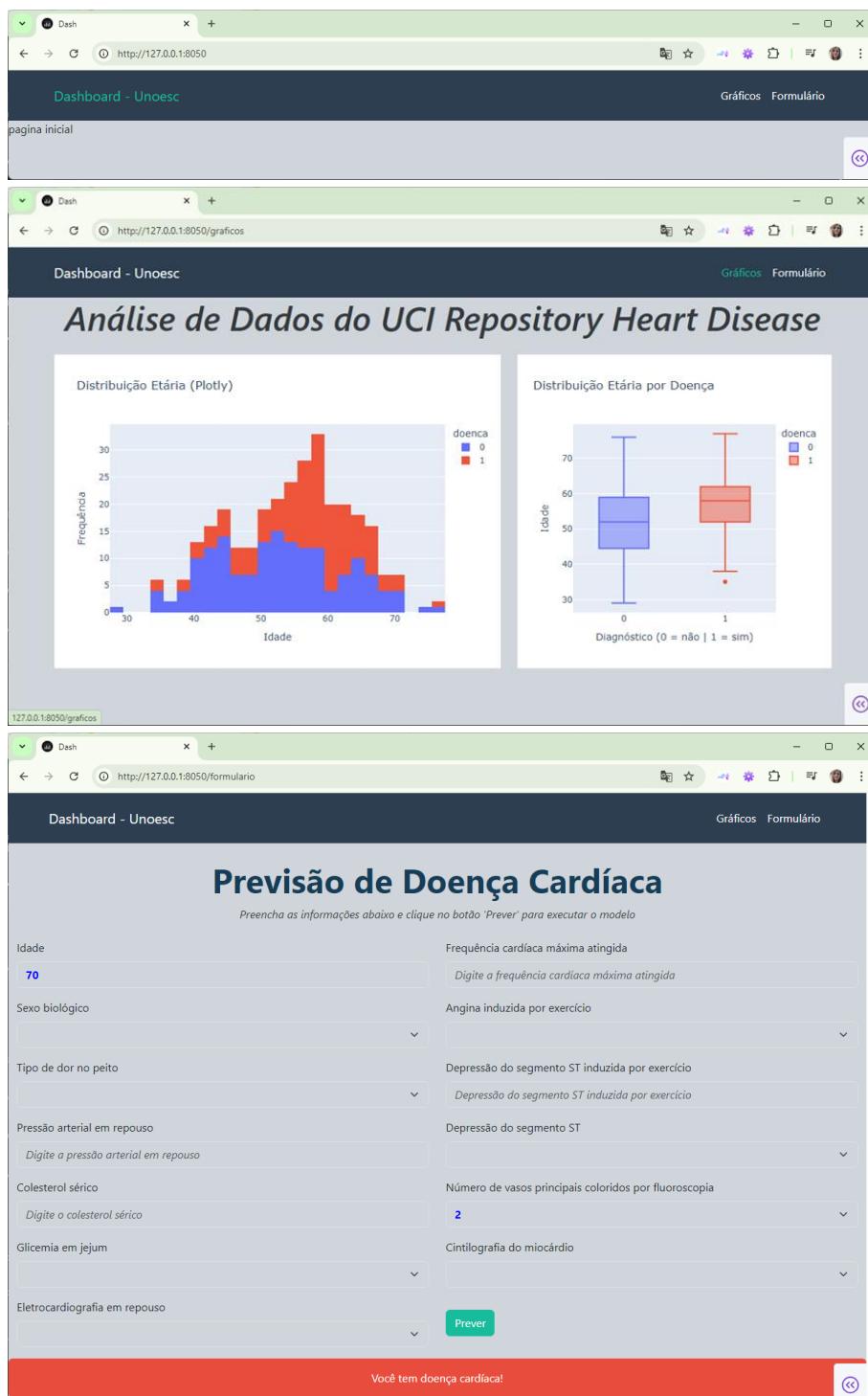
E apague as linhas:

```
app = Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
```

E:

```
app.run(debug=True)
```

Ao executar o programa principal as páginas serão as seguintes:



16 Deploy na Nuvem (Google Cloud)

16.1 Criação da Conta na Google Cloud

O último passo será o *deploy* do *dashboard*, que será feito no [Google Cloud](#). Para iniciar os seus estudos você pode acessar es [link](#) e clicar em *Comece gratuitamente*. Assim, você terá o crédito de US\$ 300 dólares fornecido pela Google para aproveitar a plataforma. Será necessário informar um número de cartão de crédito para o cadastro.

The screenshot shows the Google Cloud free trial landing page. At the top, there's a navigation bar with links for 'Visão Geral', 'Soluções', 'Produtos', 'Preços', and 'Recursos'. A search bar and links for 'Documentos', 'Suporte', and 'Fazer login' are also present. Two prominent buttons at the top right are 'Entre em contato com nossa equipe' and 'Comece gratuitamente'. Below this, a large heading reads 'Inscrição, teste e uso mensal sem custos financeiros'. Underneath are two smaller buttons: 'Comece a usar gratuitamente' and 'Entre em contato com a equipe de vendas'. The main content area features a section titled 'Três formas de começar gratuitamente' with three columns of text and links:

Crédito gratuito de US\$ 300 para novos clientes	Comece a implantar soluções pré-criadas gratuitamente	Mais de 20 produtos com Nível gratuito
Clientes novos ganham US\$ 300 em crédito para testar produtos do Google Cloud e criar uma prova de conceito. Você só vai receber uma cobrança depois que ativar a conta paga completa.	Aplique seu crédito gratuito de US\$ 300 para implantar soluções pré-criadas recomendadas pelo Google, como um site dinâmico , uma VM com balanceamento de carga , app da Web de três níveis e muito mais .	Use gratuitamente as APIs de IA , o Compute Engine , o BigQuery e outros produtos conhecidos até atingir os limites mensais , mas não cobrados do seu crédito gratuito de US\$ 300.

Clique no botão *Console* no canto superior direito ou o botão '*Acesse o console*' mais abaixo para começar a utilizar os recursos da plataforma.

The screenshot shows the Google Cloud Vertex AI landing page. The top navigation bar is identical to the previous screenshot. A large heading in the center says 'Crie aplicativos agênticos de última geração'. Below it, a subtext states: 'Com a Vertex AI, você orquestra os três pilares da IA de produção: modelos, dados e agentes.' There are two buttons at the bottom left: 'Teste a Vertex AI' and 'Entre em contato com a equipe de vendas'. To the right, there's a dark callout box with text: 'AI optimized platform', 'Open multicloud', 'Built for interoperability', and 'Assista os destaques da palestra de abertura do Next 25'. At the very bottom, there are links for 'Novidades sobre IA', 'Desenvolvedores', and 'líderes empresariais'.

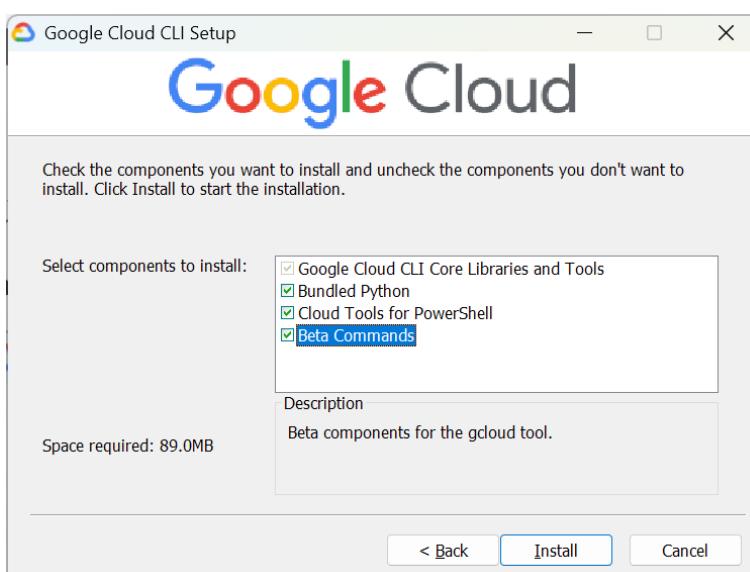
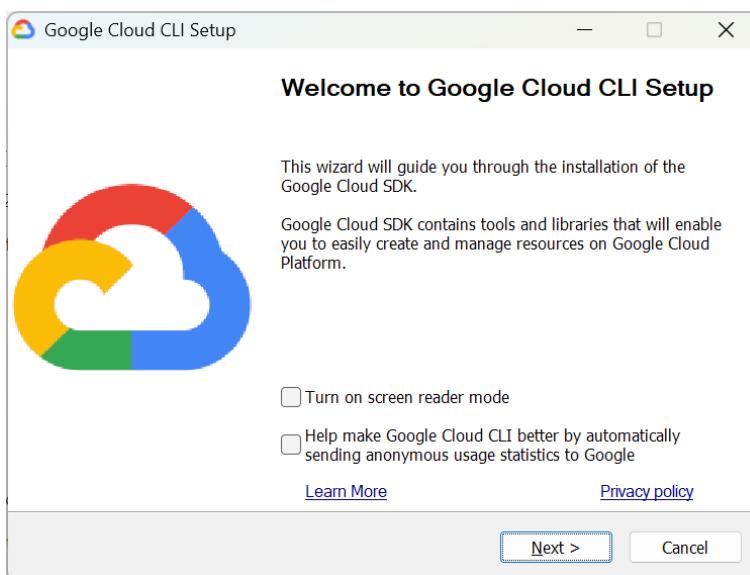
Mesmo utilizando a camada gratuita, deve-se prestar atenção nos produtos / serviços sendo utilizados. Ao receber a nota do trabalho, pode-se excluir o projeto caso haja preocupação com cobranças.

The screenshot shows the Google Cloud 'Produtos do Nível gratuito' (Free Tier Products) page. The page lists 18 different services, each with a brief description and a link to more information. The services are arranged in a grid:

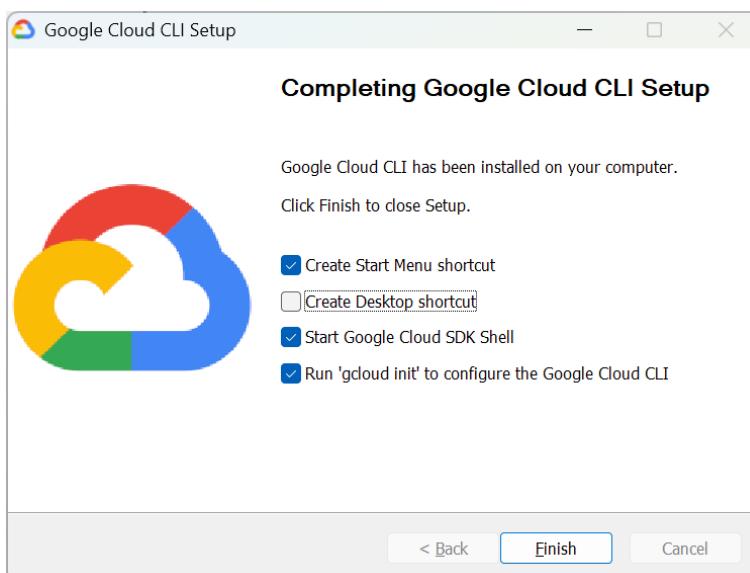
- Compute Engine**: Máquinas virtuais escalonáveis de alto desempenho. 1 instância e2-micro por mês.
- Cloud Storage**: Melhor desempenho, confiabilidade e preço para todas as necessidades de armazenamento. 5 GB por mês na classe Standard Storage.
- BigQuery**: Armazenamento de dados de análise totalmente gerenciado, capaz de processar petabytes de dados. 1 TB de consultas por mês.
- Cloud Run**: Um ambiente totalmente gerenciado para executar contêineres sem estado, criar apps ou hospedar e implantar sites. 2 milhões de solicitações por mês.
- Google Kubernetes Engine**: Orquestração de contêineres com um clique usando clusters do Kubernetes, gerenciados pelo Google. Um cluster zonal ou Autopilot por mês.
- Cloud Build**: Builds rápidos, consistentes e confiáveis no Google Cloud. 120 minutos de compilação por dia.
- Operações (anteriormente, Stackdriver)**: Monitoramento, geração de registros e diagnósticos para aplicativos no Google Cloud. Cotas mensais para geração de registros e monitoramento.
- Firestore**: Banco de dados de documentos NoSQL que simplifica o armazenamento, a sincronização e a consulta de dados para apps. 1 GB de armazenamento.
- Pub/Sub**: Serviço global para dados de streaming e mensagens confiáveis e em tempo real. 10 GB de mensagens por mês.
- Funções do Cloud Run**: Ambiente sem servidor para criar e conectar serviços em nuvem usando código. 2 milhões de invocações por mês.
- Vision AI**: Detecção de rótulos, reconhecimento óptico de caracteres, detecção facial e outros recursos. 1.000 unidades por mês.
- Speech-to-Text**: Transcrição com conversão de voz em texto, a mesma tecnologia usada nos produtos Google. 60 minutos por mês.
- API Natural Language**: Gere insights de textos não estruturados usando o machine learning do Google. 5.000 unidades por mês.
- Cloud KMS Autokey**: Receba as chaves certas do Cloud KMS on demand para ter um alinhamento consistente com as práticas de criptografia recomendadas. 100 versões ativas de chaves e 10.000 operações de chaves por mês.
- API Video Intelligence**: Modelos de ML pré-treinados que reconhecem objetos, lugares e ações em vídeos armazenados e via streaming. 1.000 unidades por mês.
- Fluxos de trabalho**: Execute sequências de chamadas de serviços totalmente gerenciadas no Google Cloud e em qualquer API HTTP. 5 mil etapas internas gratuitas por mês.
- Cloud Source Repositories**: Terá vários repositórios Git particulares hospedados no Google Cloud. Acesso gratuito para até cinco usuários.
- Google Cloud Marketplace**: Soluções click-to-deploy de parceiros do Google Cloud prontas para produção. Testes gratuitos de apps e serviços selecionados.
- Secret Manager**: Armazene com segurança chaves de API, senhas, certificados e outros dados confidenciais. Seis versões de secrets por mês.
- Cloud Shell**: Ambiente de operações e desenvolvimento on-line acessível em qualquer lugar com seu navegador.

16.2 Instalação da GCLOUD (Google Cloud CLI)

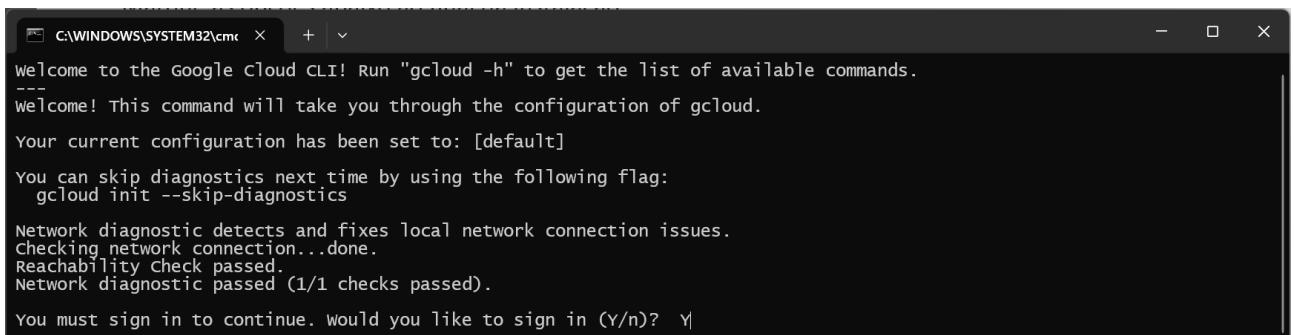
A [Google Cloud CLI](#) (CLI gcloud) é um conjunto de ferramentas para criar e gerenciar recursos do Google Cloud. É possível usar essas ferramentas para realizar muitas tarefas comuns da plataforma pela linha de comando ou por *scripts* e outras automações. Baixe e execute o instalador do Google Cloud CLI (*Command Line Interface*) neste [link](#).



Marque as opções abaixo ao final da instalação:

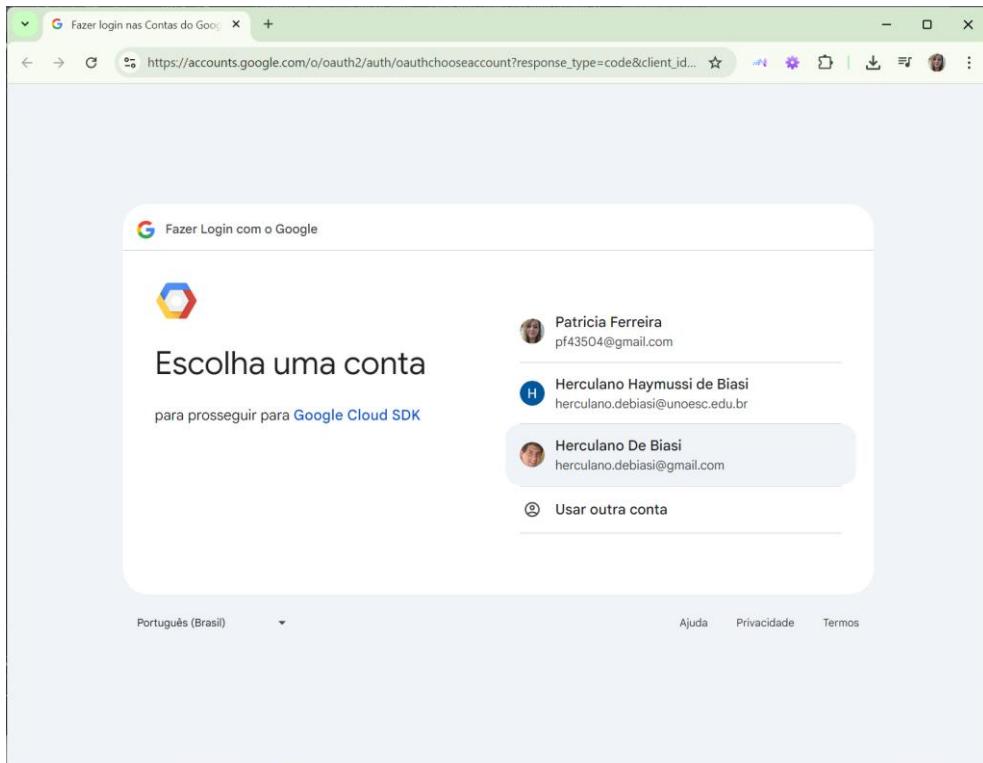


Responda **Y** para proceder com o processo de autenticação:

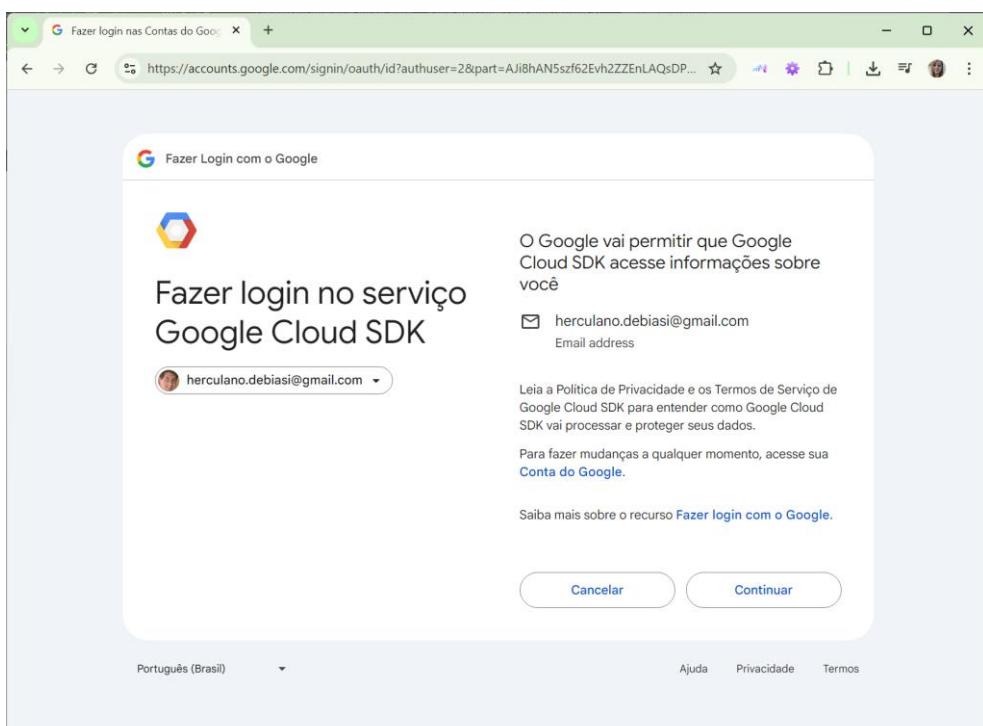


```
C:\WINDOWS\SYSTEM32\cmd > + -  
Welcome to the Google Cloud CLI! Run "gcloud -h" to get the list of available commands.  
---  
welcome! This command will take you through the configuration of gcloud.  
Your current configuration has been set to: [default]  
You can skip diagnostics next time by using the following flag:  
  gcloud init --skip-diagnostics  
Network diagnostic detects and fixes local network connection issues.  
Checking network connection...done.  
Reachability check passed.  
Network diagnostic passed (1/1 checks passed).  
You must sign in to continue. Would you like to sign in (Y/n)? Y|
```

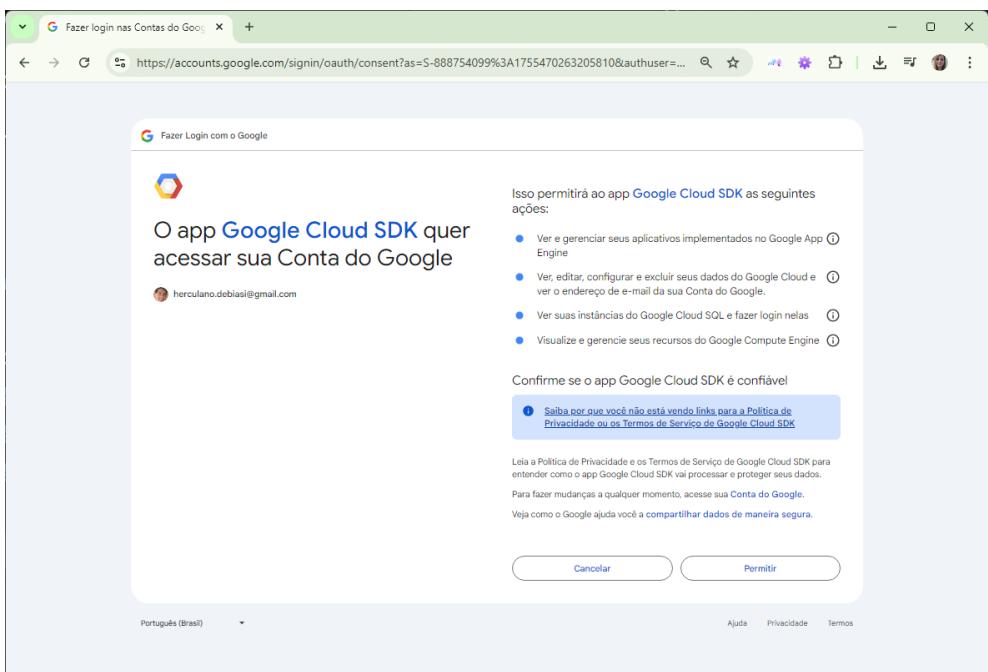
Escolha sua conta **PESSOAL** (não acadêmica) para autenticação:



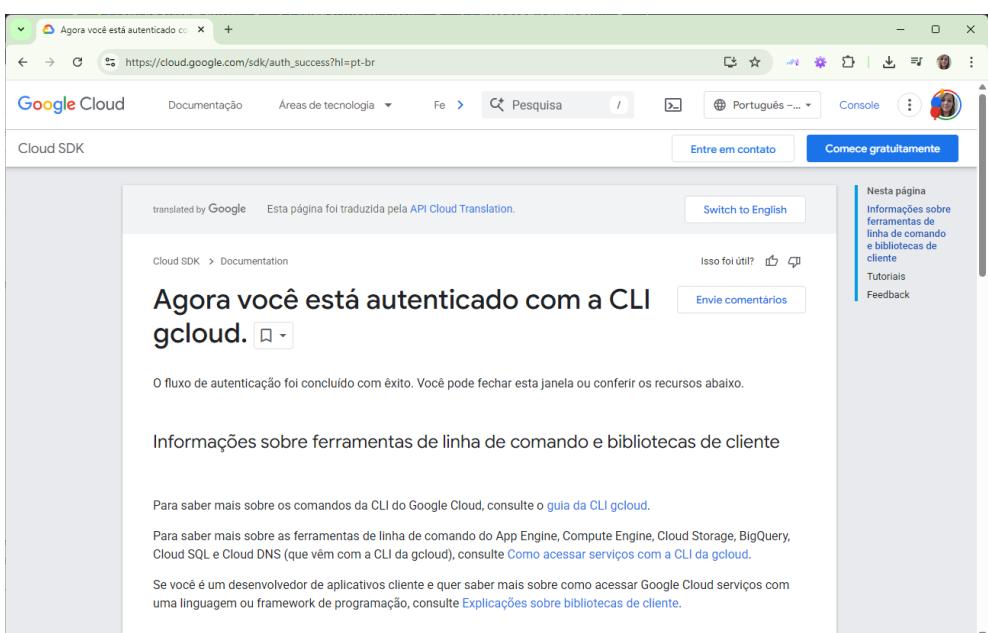
Clique em *Continuar*:



Clique em Permitir:



Tela mostrando que a autenticação foi realizada com sucesso:



IMPORTANTE: Caso você fique muito tempo sem trabalhar com o projeto e seja deslogado, deverá executar usando os comandos abaixo para logar e definir o projeto padrão para as operações da gcloud. Isso significa que todos os comandos subsequentes da CLI gcloud, que precisam de um projeto, usarão o projeto especificado por este comando como o projeto padrão, a menos que um projeto diferente seja explicitamente especificado no comando.

- gcloud auth login
- gcloud config set project dashapp-hdb

Verificando a conta ativa:

- gcloud auth list

```
Google Cloud SDK Shell
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud auth list
Credentialed Accounts
ACTIVE  ACCOUNT
*      herculano.debiasi@gmail.com
       herculano.debiasi@unoesc.edu.br
To set the active account, run:
$ gcloud config set account `ACCOUNT`
```

16.3 Criação do Projeto na Google Cloud

Google Cloud SDK Shell:

```
Google Cloud SDK Shell
Welcome to the Google Cloud CLI! Run "gcloud -h" to get the list of available commands.
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Responda *Y* ou o número indicado para criar um novo projeto:

```
You are signed in as: [herculano.debiasi@unoesc.edu.br].
This account has no projects.
would you like to create one? (y/n)? Y

You are signed in as: [herculano.debiasi@gmail.com].
Pick cloud project to use:
[1] chatbotpizza-xmgy
[2] clone-whatsapp-54337
[3] cloud-e-Iot-2023-herculano
[4] pizzaria-renx
[5] projeto-teste-bb10b
[6] teste-api-v3-do-youtube
[7] todolist-herculano
[8] webapp-java-33412
[9] Enter a project ID
[10] Create a new project
Please enter numeric choice or text value (must exactly match list item): 10
```

Insira um nome para o projeto – ele NÃO poderá ser alterado posteriormente. O nome deve ser único globalmente. No exemplo abaixo foi inserido um nome já existente - dashapp - causando o erro: '*Requested entity already exists*':

```
Enter a Project ID. Note that a Project ID CANNOT be changed later.
Project IDs must be 6-30 characters (lowercase ASCII, digits, or
hyphens) in length and start with a lowercase letter. dashapp
WARNING: Project creation failed: HttpError accessing <https://clouresourcemanager.googleapis.com/v1/projects?alt=json>; response: <{'vary': 'Origin, X-Origin, Referer', 'content-type': 'application/json; charset=UTF-8', 'content-encoding': 'gzip', 'date': 'Sun, 17 Aug 2025 22:43:21 GMT', 'server': 'ESF', 'x-xss-protection': '0', 'x-frame-options': 'SAMEORIGIN', 'x-content-type-options': 'nosniff', 'server-timing': 'qfet4t7; dur=792', 'alt-svc': 'h3=:443'; ma=2592000,h3-29=:443'; ma=2592000', 'transfer-encoding': 'chunked', 'status': 409}>, content <{
  "error": {
    "code": 409,
    "message": "Requested entity already exists",
    "status": "ALREADY_EXISTS"
  }
}
Please make sure to create the project [dashapp] using
  $ gcloud projects create dashapp
or change to another project using
  $ gcloud config set project <PROJECT ID>
The Google Cloud CLI is configured and ready to use!

* Commands that require authentication will use herculano.debiasi@gmail.com by default
Run `gcloud help config` to learn how to change individual settings

This gcloud configuration is called [default]. You can create additional configurations if you work
with multiple accounts and/or projects.
Run `gcloud topic configurations` to learn more.

Some things to try next:
* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to get help on any gcloud command.
* Run `gcloud topic --help` to learn about advanced features of the CLI like arg files and output fo
rmatting
* Run `gcloud cheat-sheet` to see a roster of go-to `gcloud` commands.

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Como indicado na tela acima, uma nova tentativa de criação de projeto deverá ser feita com um novo nome.

Para criar o projeto, use o seguinte comando:

➤ `gcloud projects create dashapp-hdb`

Dessa vez o processo foi bem sucedido:

```
Google Cloud SDK Shell
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud projects create dashapp-hdb
Create in progress for [https://clouresourcemanager.googleapis.com/v1/projects/dashapp-hdb].
Waiting for [operations/create_project_global.8033593451170310511] to finish...done.
Enabling service [cloudfoundry.googleapis.com] on project [dashapp-hdb]...
Operation "operations/acat.p2-416128962446-5b24fc3f-33f1-49f2-a568-6fde07cda256" finished successfully.

Updates are available for some Google Cloud CLI components. To install them,
please run:
  $ gcloud components update

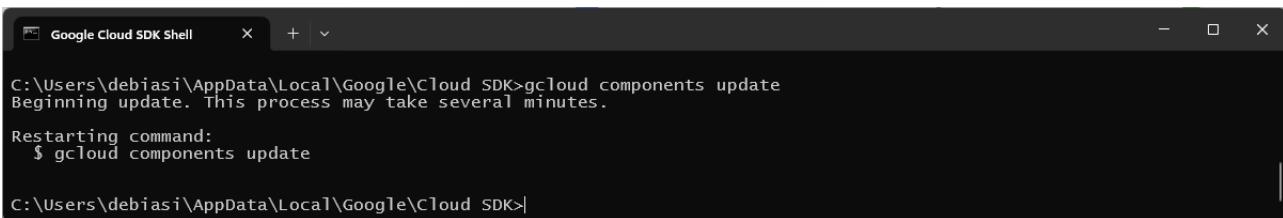
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

ATENÇÃO: Como o nome *dashapp-hdb* já foi escolhido pelo professor, você acadêmico terá que escolher outro.

É solicitada a instalação de atualizações:

```
➤ gcloud components update
```

Responda *Y* na tela que irá aparecer:



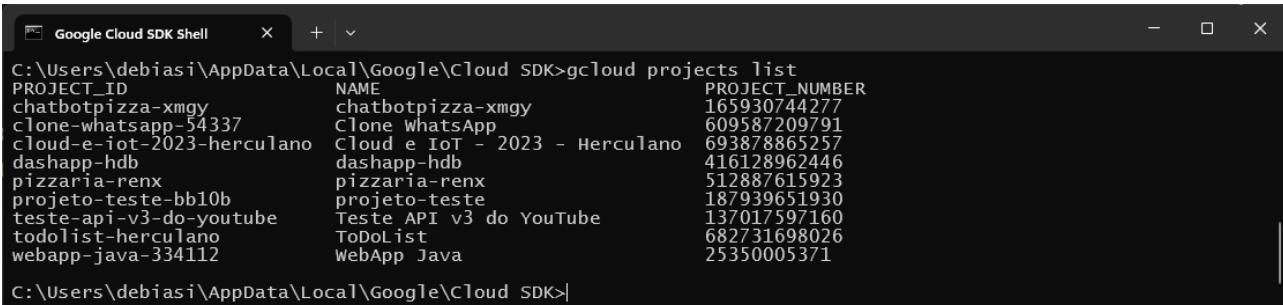
```
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud components update
Beginning update. This process may take several minutes.

Restarting command:
$ gcloud components update

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Verificando seus projetos na Google Cloud:

```
➤ gcloud projects list
```

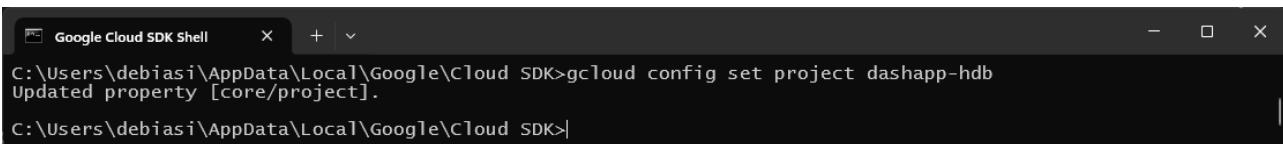


PROJECT_ID	NAME	PROJECT_NUMBER
chatbotpizza-xmgy	chatbotpizza-xmgy	165930744277
clone-whatsapp-54337	Clone WhatsApp	609587209791
cloud-e-iot-2023-herculano	Cloud e IoT - 2023 - Herculano	693878865257
dashapp-hdb	dashapp-hdb	416128962446
pizzaria-renx	pizzaria-renx	512887615923
projeto-teste-bb10b	projeto-teste	187939651930
teste-api-v3-do-youtube	Teste API v3 do YouTube	137017597160
todolist-herculano	ToDoList	682731698026
webapp-java-334112	WebApp Java	25350005371

```
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Definindo o projeto como *default*:

```
➤ gcloud config set project dashapp-hdb
```

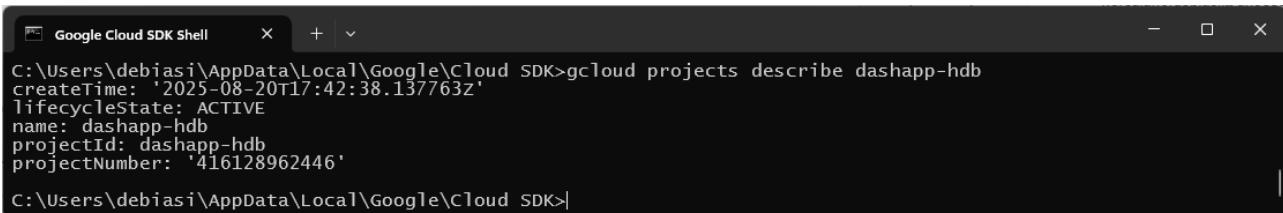


```
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud config set project dashapp-hdb
Updated property [core/project].

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Obtendo mais detalhes sobre um projeto:

```
➤ gcloud projects describe dashapp-hdb
```

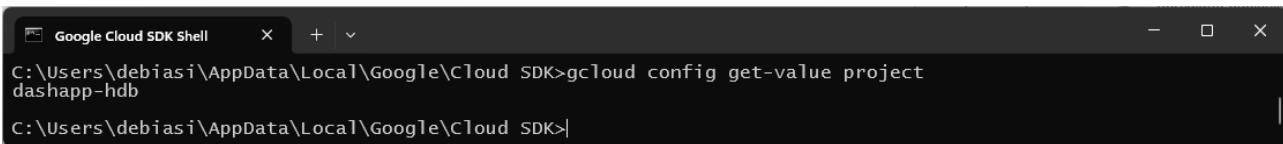


```
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud projects describe dashapp-hdb
createTime: '2025-08-20T17:42:38.137763Z'
lifecycleState: ACTIVE
name: dashapp-hdb
projectId: dashapp-hdb
projectNumber: '416128962446'

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

Verificando o projeto ativo:

```
➤ gcloud config get-value project
```



```
C:\Users\debiasi\AppData\Local\Google\Cloud SDK>gcloud config get-value project
dashapp-hdb

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

O comando abaixo também fornece várias informações, incluindo a conta e projeto correntes:

```
➤ gcloud info
```



```
Account: [herculano.debiasi@gmail.com]
Project: [dashapp-hdb]
Universe Domain: [googleapis.com]

C:\Users\debiasi\AppData\Local\Google\Cloud SDK>
```

É possível selecionar o projeto na Google Cloud, na aba *Todos*:

The screenshot shows the Google Cloud Console interface. A modal window titled "Selecionar um projeto" (Select a project) is open, listing various projects under the "Todos" tab. The projects listed include "Clone WhatsApp", "Cloud e IoT - 2023 - Herculano", "Teste API v3 do YouTube", "ToDoList", "WebApp Java", "chatbotpizza-xmgy", "dashapp-hdb", "pizzaria-renx", and "projeto-teste". Each entry shows the project name, type (Projeto or Organização), ID, and a star icon for favoriting. The "Novo projeto" (New project) button is visible at the top right of the modal.

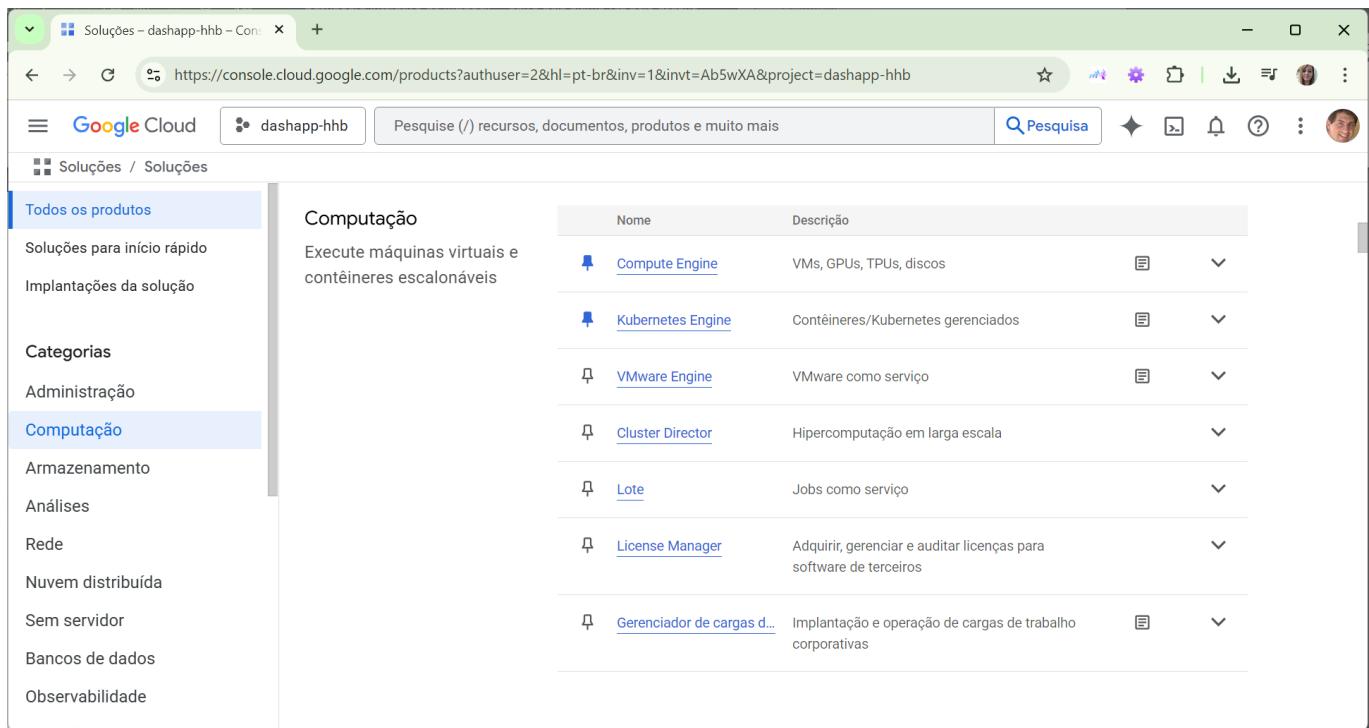
Página do projeto.

The screenshot shows the Google Cloud Console interface for the project "dashapp-hdb". The main header indicates "Olá!" and "Você está trabalhando em dashapp-hdb". Below the header, it shows the project number "Número do projeto: 416128962446" and the project ID "ID do projeto: dashapp-hdb". The "Cloud Hub" tab is selected. The "Acesso rápido" (Quick access) section contains links for "API APIs e serviços", "IAM e administrador", "Faturamento", "Compute Engine", "Cloud Storage", "BigQuery", "Rede VPC", and "Kubernetes Engine". A "Ver todos os produtos" (View all products) button is also present. A sidebar on the right provides a "Teste o chat do Gemini Cloud Assist" (Test the Gemini Cloud Assist chat) feature with a "Inicie um chat agora" (Start a chat now) button.

Anote o ID e o número do seu projeto, semelhantes aos vistos acima:

- **ID:** dashapp-hdb
- **Número:** 416128962446

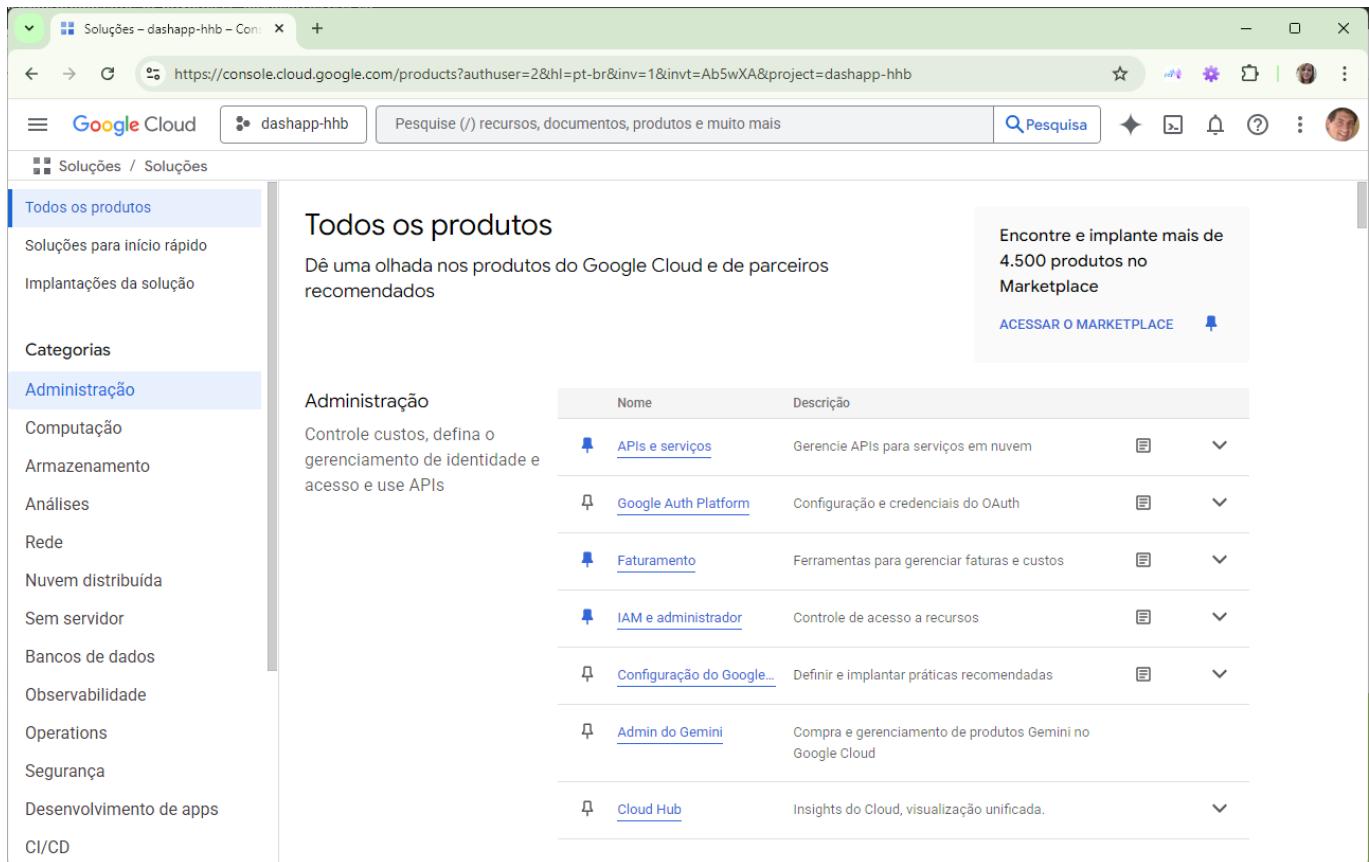
Clique em *Ver todos os produtos* mostrado logo acima. Existe uma quantidade enorme de produtos e serviços oferecidos pela Google Cloud, como as de computação mostradas na tela abaixo:



The screenshot shows the Google Cloud Console interface. The left sidebar has a 'Todos os produtos' section with categories like 'Soluções para início rápido' and 'Implantações da solução'. Below that is a 'Categorias' section with 'Administração', 'Computação', 'Armazenamento', 'Análises', 'Rede', 'Nuvem distribuída', 'Sem servidor', 'Bancos de dados', and 'Observabilidade'. The main content area is titled 'Computação' and describes executing virtual machines and managed containers. It lists several services: Compute Engine (VMs, GPUs, TPUs, disks), Kubernetes Engine (managed Kubernetes), VMware Engine (VMware as a service), Cluster Director (large-scale hyper-computing), Lote (jobs as a service), License Manager (licensing management), and Gerenciador de cargas (workload management).

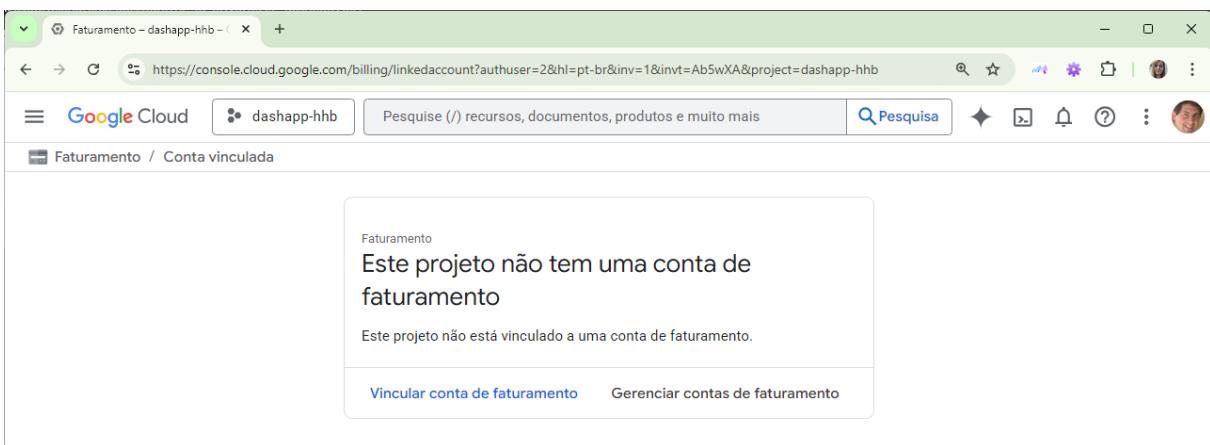
16.4 Conta de Faturamento

Volte para o projeto no Google Console e clique no link *Faturamento*.

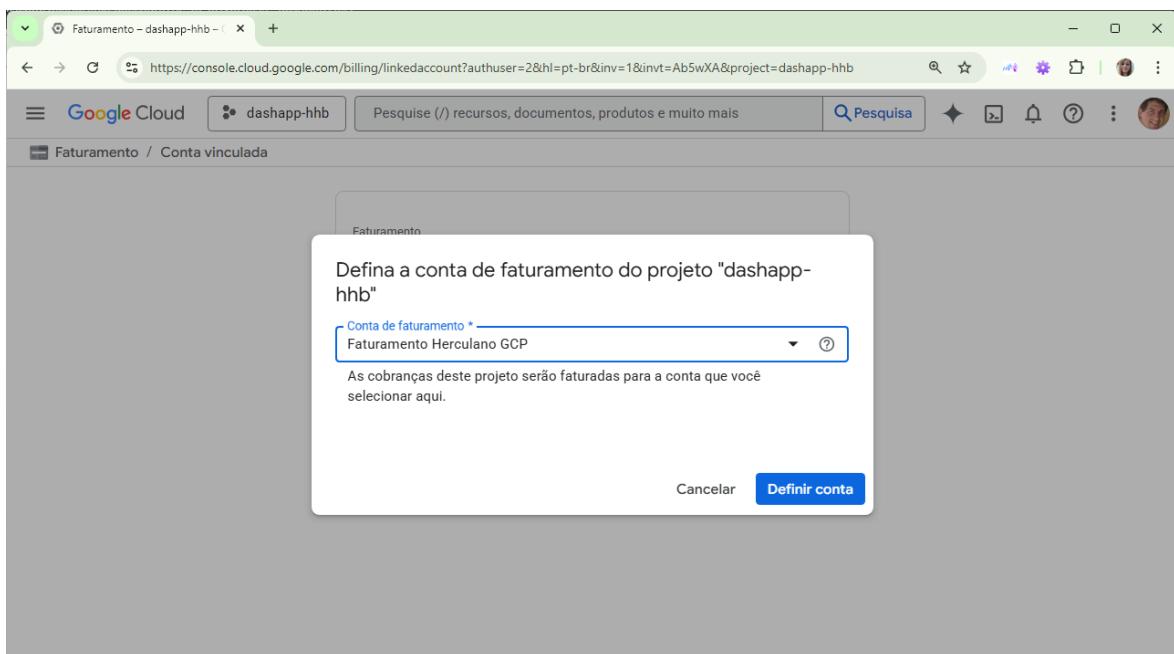


The screenshot shows the Google Cloud Console interface. The left sidebar has a 'Todos os produtos' section with categories like 'Soluções para início rápido' and 'Implantações da solução'. Below that is a 'Categorias' section with 'Administração', 'Computação', 'Armazenamento', 'Análises', 'Rede', 'Nuvem distribuída', 'Sem servidor', 'Bancos de dados', 'Observabilidade', 'Operations', 'Segurança', 'Desenvolvimento de apps', and 'CI/CD'. The main content area is titled 'Todos os produtos' and says 'Dê uma olhada nos produtos do Google Cloud e de parceiros recomendados'. It features a callout for the Marketplace with text 'Encontre e implante mais de 4.500 produtos no Marketplace' and a 'ACESSAR O MARKETPLACE' button. The main list under 'Administração' includes: APIs e serviços (API management), Google Auth Platform (OAuth configuration), Faturamento (Billing tools), IAM e administrador (Access control), Configuração do Google... (Google configuration), Admin do Gemini (Gemini product management), and Cloud Hub (Cloud insights).

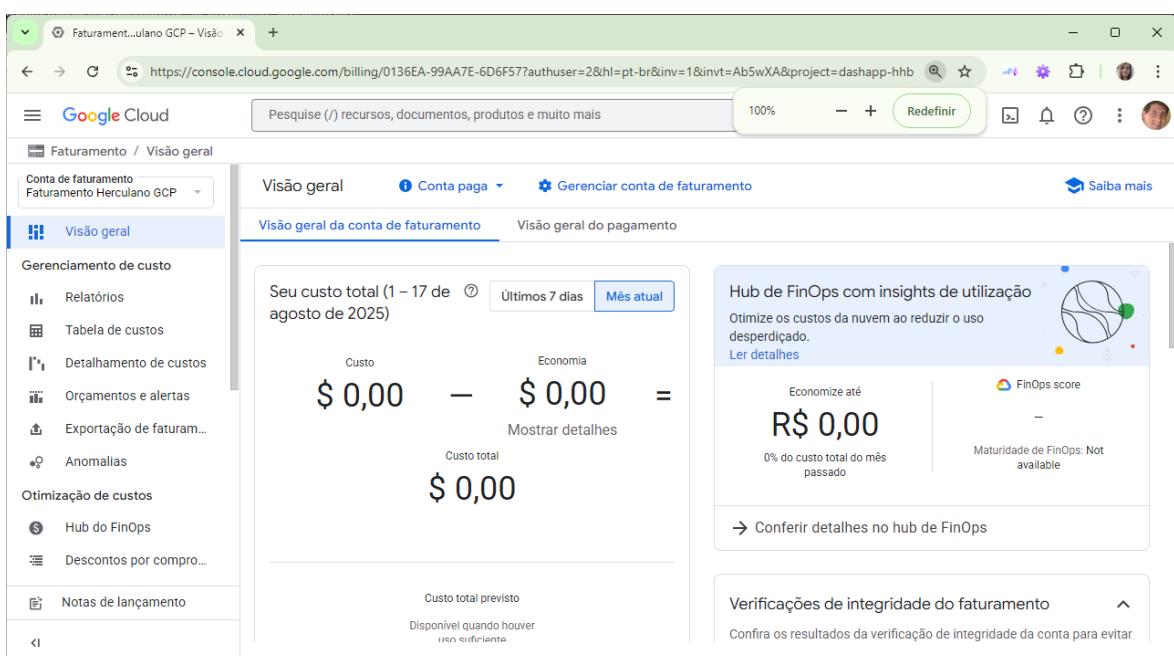
Clique em *Vincular conta de faturamento*:



Escolha a conta e clique em *Definir conta*:



Caso seja necessária alguma cobrança para executar o aplicativo, ela será feita por meio dessa conta vinculada a um cartão de crédito:



16.5 Habilitação de APIs

Clique neste [link](#) para abrir o guia ‘Implantar um aplicativo conteinerizado no Cloud Run usando o Cloud Build’:

The screenshot shows the 'Guia de início rápido: implantar' page from the Google Cloud Build documentation. The main content area is titled 'Antes de começar' (Before you begin) and lists three steps: 1. Select or create a Google Cloud project, 2. Verify that billing is enabled for your Google Cloud project, and 3. Enable the Cloud Build, Cloud Run, Artifact Registry, and Compute Engine APIs. A sidebar on the left contains links for 'Configurar', 'Criar um arquivo de configuração de build', 'Esquema do arquivo de configuração do build', 'Desenvolvedores de nuvens', 'Usar builders de contribuições da comunidade e personalizados', 'Substituir valores de variáveis', 'Executar scripts bash', 'Interagir com imagens do Docker Hub', 'Configurar a ordem das etapas de build', 'Transmitir dados entre etapas de build', and 'Usar vinculações de payload e'. A sidebar on the right lists 'Nesta página' with links to 'Antes de começar', 'Conceder permissões', 'Implantar uma imagem pré-criada', 'Executar a imagem implantada', and 'A seguir'.

Clique no botão *Enable the APIs* mostrado acima, confirme se é o projeto correto e então clique em *Avançar*:

The screenshot shows the 'Ativar acesso a APIs – APIs e serviços' page in the Google Cloud console. It displays two steps: 1. Confirmar projeto (checkbox checked) and 2. Ativar APIs (checkbox selected). Below these steps, it says 'As mudanças vão ser feitas no projeto dashapp-hdb. Se esse não for o projeto que você quer usar, selecione ou crie outro com o seletor acima.' and has a 'Avançar' button.

Clique em *Ativar*:

The screenshot shows the same 'Ativar acesso a APIs – APIs e serviços' page, but now the 'Ativar APIs' step is selected. It displays a list of APIs being activated: Cloud Build API, Cloud Run Admin API, Artifact Registry API, and Compute Engine API. There is also an 'Ativar' button at the bottom.

Resultado:

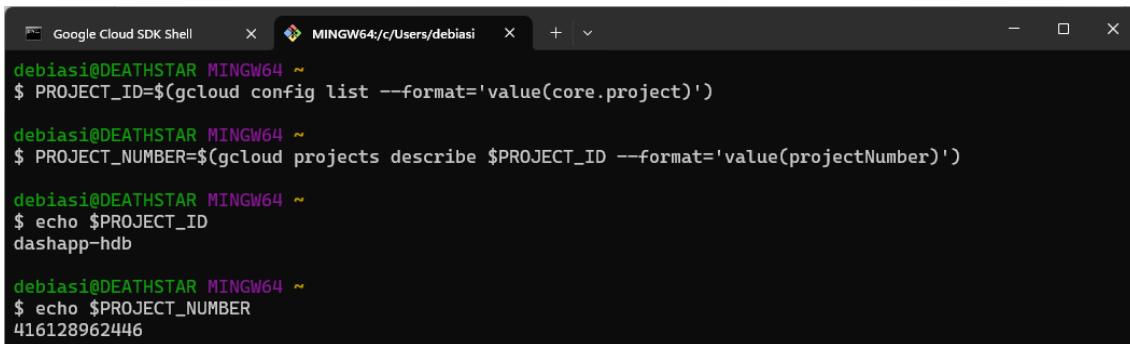
The screenshot shows the final state of the 'Ativar acesso a APIs – APIs e serviços' page. Both the 'Confirmar projeto' and 'Ativar APIs' steps have checkboxes checked. The 'Ativar APIs' step is highlighted. Below the steps, it says 'Você ativou:' followed by the list of activated APIs: Cloud Build API, Cloud Run Admin API, Artifact Registry API, and Compute Engine API. There is also an 'Ativar' button at the bottom.

16.6 Configuração Básica

Os próximos comandos deverão ser feitos em algum terminal estilo Linux, como o GitBash ou o WSL. Se não tiver o GitBash no seu sistema, instale-o clicando neste [link](#).

Execute os comandos abaixo para realizar configurações básicas do projeto:

- ```
> PROJECT_ID=$(gcloud config list --format='value(core.project)')
> PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format='value(projectNumber)')
> echo $PROJECT_ID
> echo $PROJECT_NUMBER
```



The screenshot shows a terminal window with two tabs: "Google Cloud SDK Shell" and "MINGW64:/c/Users/debiasi". The user has run several commands to retrieve their Google Cloud project information. The output is as follows:

```
debiasi@DEATHSTAR MINGW64 ~
$ PROJECT_ID=$(gcloud config list --format='value(core.project)')

debiasi@DEATHSTAR MINGW64 ~
$ PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format='value(projectNumber)')

debiasi@DEATHSTAR MINGW64 ~
$ echo $PROJECT_ID
dashapp-hhb

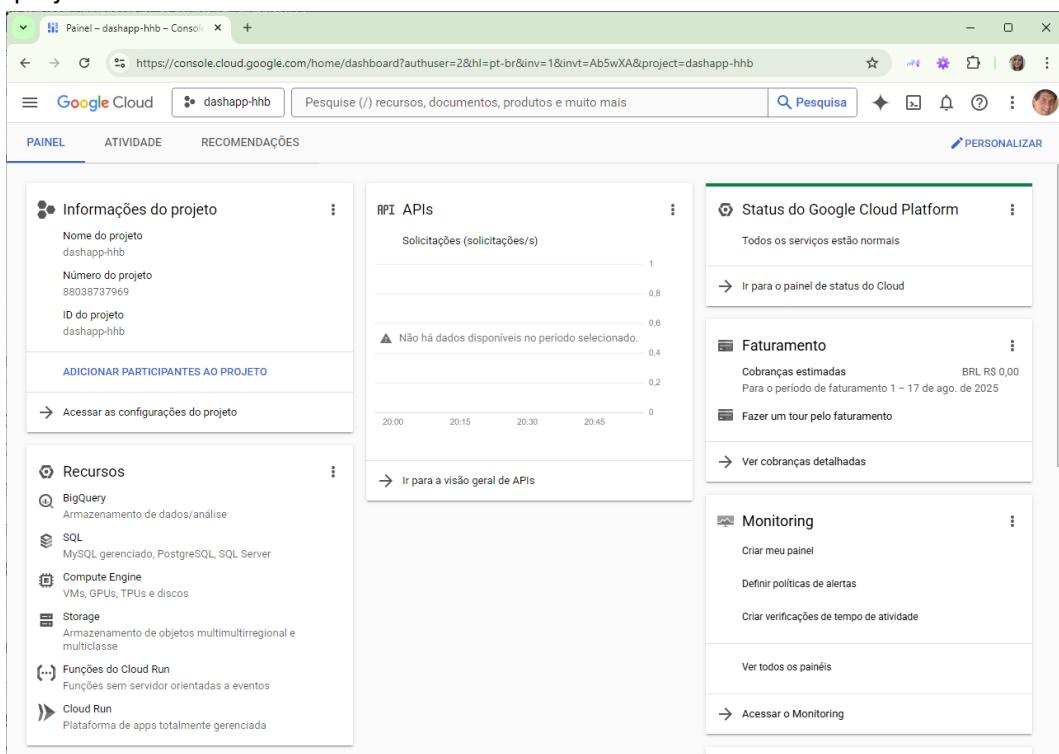
debiasi@DEATHSTAR MINGW64 ~
$ echo $PROJECT_NUMBER
416128962446
```

Verifique se as informações correspondem às anotadas na página anterior do roteiro.

Execute agora os seguintes comandos, um por vez, verificando se não houve nenhum erro:

- ```
> gcloud projects add-iam-policy-binding $PROJECT_ID \  
  --member=serviceAccount:$PROJECT_ID \  
  --format="value(projectNumber)" -compute@developer.gserviceaccount.com \  
  --role=roles/run.admin  
> gcloud projects add-iam-policy-binding $PROJECT_ID \  
  --member=serviceAccount:$PROJECT_ID \  
  --format="value(projectNumber)" -compute@developer.gserviceaccount.com \  
  --role="roles/storage.objectUser"  
> gcloud iam service-accounts add-iam-policy-binding $(gcloud projects describe $PROJECT_ID \  
  --format="value(projectNumber)") -compute@developer.gserviceaccount.com \  
  --member=serviceAccount:$PROJECT_ID \  
  --format="value(projectNumber)" -compute@developer.gserviceaccount.com \  
  --role="roles/iam.serviceAccountUser" \  
  --project=$PROJECT_ID
```

Visão geral do projeto:



The screenshot shows the Google Cloud Platform dashboard for the project "dashapp-hhb". The dashboard is divided into several sections:

- PAINEL**: Shows basic project information (Nome do projeto: dashapp-hhb, Número do projeto: 88038737969, ID do projeto: dashapp-hhb), a list of participants, and links to access project configurations and general API views.
- ATIVIDADE**: Shows API activity with a chart titled "Solicitações (solicitações/s)" showing a single request at 20:00.
- RECOMENDAÇÕES**: Shows the status of Google Cloud services ("Todos os serviços estão normais"), estimated billing ("Cobranças estimadas: BRL R\$ 0,00"), a tour of the billing section, and detailed billing reports.
- Informações do projeto**: Detailed project info including name, number, ID, and participant addition.
- Recursos**: Lists BigQuery, SQL, Compute Engine, Storage, Cloud Functions, and Cloud Run services.
- APIs**: Shows API activity with a chart showing zero requests.
- Status do Google Cloud Platform**: Overall service status.
- Faturamento**: Billing summary and tour link.
- Monitoring**: Monitoring setup options.

16.7 Criação da Imagem Docker

O objetivo agora será criar uma máquina-virtual (VM) contendo um sistema operacional Linux, Python e demais bibliotecas, além da aplicação em si. Além disso será preciso configurar o *firewall* da VM para abrir uma porta específica para acessar a aplicação. Deverá ser disponibilizado um endereço IP ou domínio para que se possa acessar a aplicação. Uma estratégia para fazer isso é criar um ‘contêiner’ com tudo o necessário.

Para a criação do contêiner será utilizada a ferramenta Docker. As instruções de configuração e do que será instalado na imagem devem ser feitas no arquivo *Dockerfile*.

Crie o arquivo chamado *Dockerfile* na pasta do projeto (*c:\programas\ml*):

```
FROM python:3.13.5-slim-bullseye

ENV APP_HOME /app
WORKDIR $APP_HOME

COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

ENV PORT 8080
EXPOSE 8080

CMD ["python", "main.py"]
```

As instruções do arquivo informam o seguinte:

- Versão do Python a ser instalada
- Diretório de trabalho */app*, que é onde estará a aplicação, informado através da variável de ambiente *APP_HOME* é a */app* e da configuração *WORKDIR \$APP_HOME*
- Cópia do arquivo de requisitos do Python para a imagem
- Execução do *pip* para instalação dos requisitos
- Cópia do restante dos arquivos
- Exposição da porta 8080 do servidor para acesso externo
- Execução do servidor dentro do contêiner

Antes de gerar a imagem, altere a linha do arquivo *main.py*:

```
app.run(debug=False, port=8080, host='0.0.0.0')
```

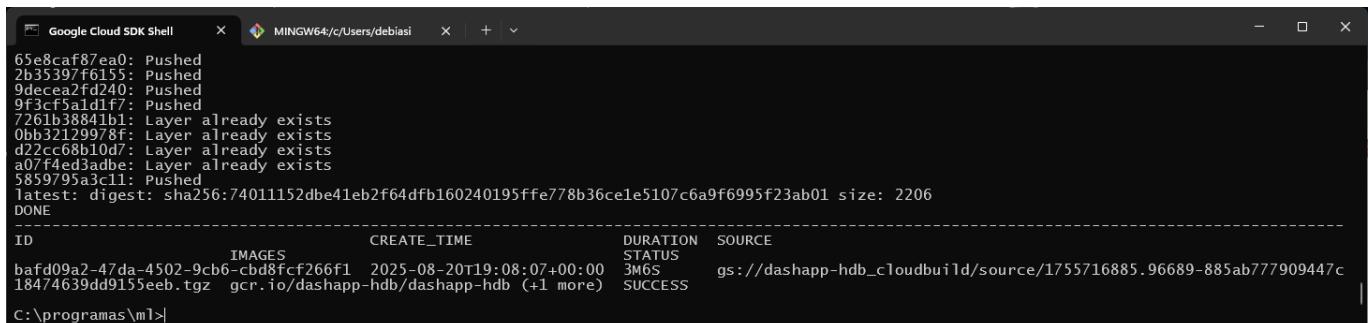
Isso informa que a porta é a 8080, por enquanto (*port=8080*) e que o host aceita requisição de todos os lugares da internet (*host='0.0.0.0'*).

Com o *prompt* do Google Cloud execute os comandos abaixo, adaptando para o **NOME** para o de seu projeto:

```
> cd \programas\ml
> gcloud builds submit --tag gcr.io/dashapp-hdb/dashapp-hdb --project=dashapp-hdb
```

Este processo deverá levar alguns minutos.

Imagem gerada com sucesso:



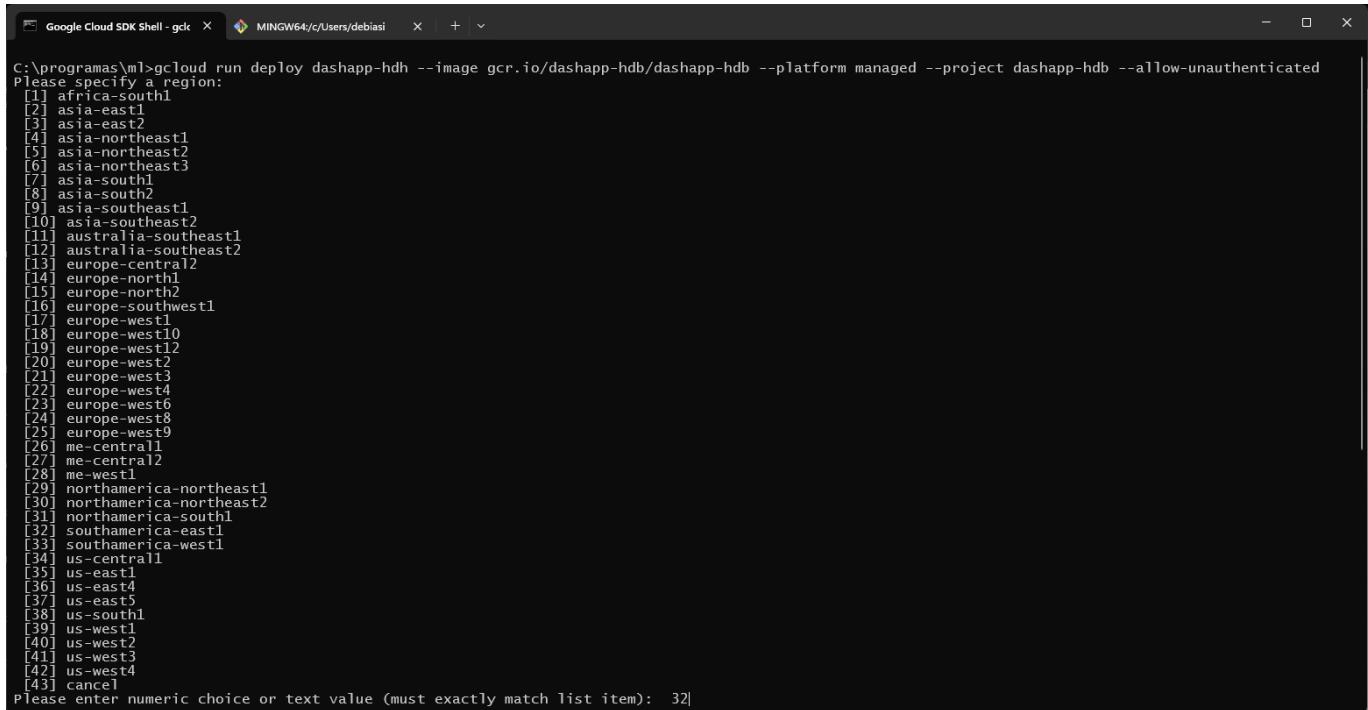
```
Google Cloud SDK Shell
65e8caf87ea0: Pushed
2b35397f6155: Pushed
9decea2fd240: Pushed
9f3cf5a1d1f7: Pushed
7261b38841b1: Layer already exists
0bb32129978f: Layer already exists
d22cc68b10d7: Layer already exists
a07f4ed3adbe: Layer already exists
5859795a3c11: Pushed
latest: digest: sha256:74011152dbe41eb2f64dfb160240195ffe778b36ce1e5107c6a9f6995f23ab01 size: 2206
DONE
ID          IMAGES      CREATE_TIME    DURATION  STATUS   SOURCE
baf09a2-47da-4502-9cb6-cbd8fcf266f1  2025-08-20T19:08:07+00:00  3M6S  SUCCESS  gs://dashapp-hdb_cloudbuild/source/1755716885.96689-885ab777909447c
C:\programas\ml>
```

16.8 Deploy (Publicação) da Imagem

Com o *prompt* do Google Cloud execute os comandos abaixo, adaptando para o **NOME** para o de seu projeto:

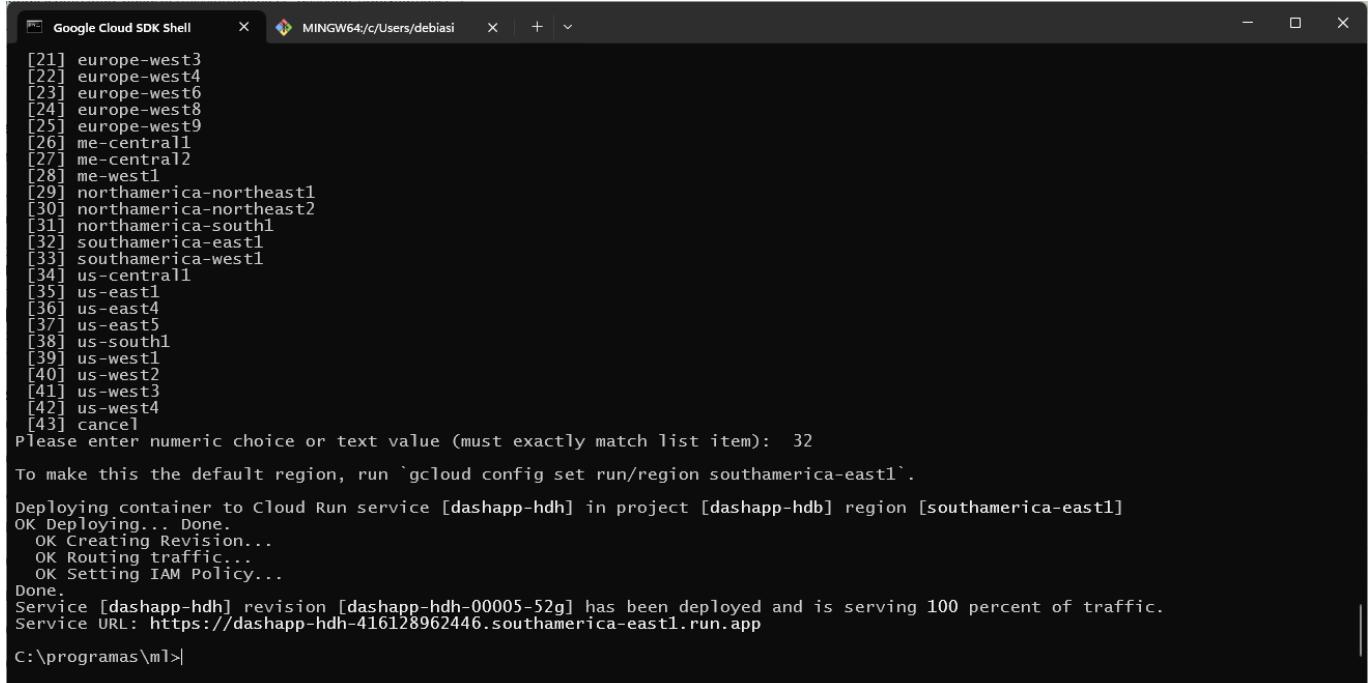
```
> gcloud run deploy dashapp-hdh --image gcr.io/dashapp-hdb/dashapp-hdb --platform managed --project dashapp-hdb --allow-unauthenticated
```

Escolha a região correspondente à *southamerica-east1*:



```
C:\programas\ml>gcloud run deploy dashapp-hdh --image gcr.io/dashapp-hdb/dashapp-hdb --platform managed --project dashapp-hdb --allow-unauthenticated
Please specify a region:
[1] africa-south1
[2] asia-east1
[3] asia-east2
[4] asia-northeast1
[5] asia-northeast2
[6] asia-northeast3
[7] asia-south1
[8] asia-south2
[9] asia-southeast1
[10] asia-southeast2
[11] australia-southeast1
[12] australia-southeast2
[13] europe-central1
[14] europe-north1
[15] europe-north2
[16] europe-southwest1
[17] europe-west1
[18] europe-west10
[19] europe-west2
[20] europe-west3
[21] europe-west4
[22] europe-west6
[23] europe-west8
[24] europe-west9
[25] europe-west9
[26] me-central1
[27] me-central12
[28] me-west1
[29] northamerica-northeast1
[30] northamerica-northeast2
[31] northamerica-south1
[32] southamerica-east1
[33] southamerica-west1
[34] us-central1
[35] us-east1
[36] us-east4
[37] us-east5
[38] us-south1
[39] us-west1
[40] us-west2
[41] us-west3
[42] us-west4
[43] cancel
Please enter numeric choice or text value (must exactly match list item): 32|
```

Deploy efetuado com sucesso:

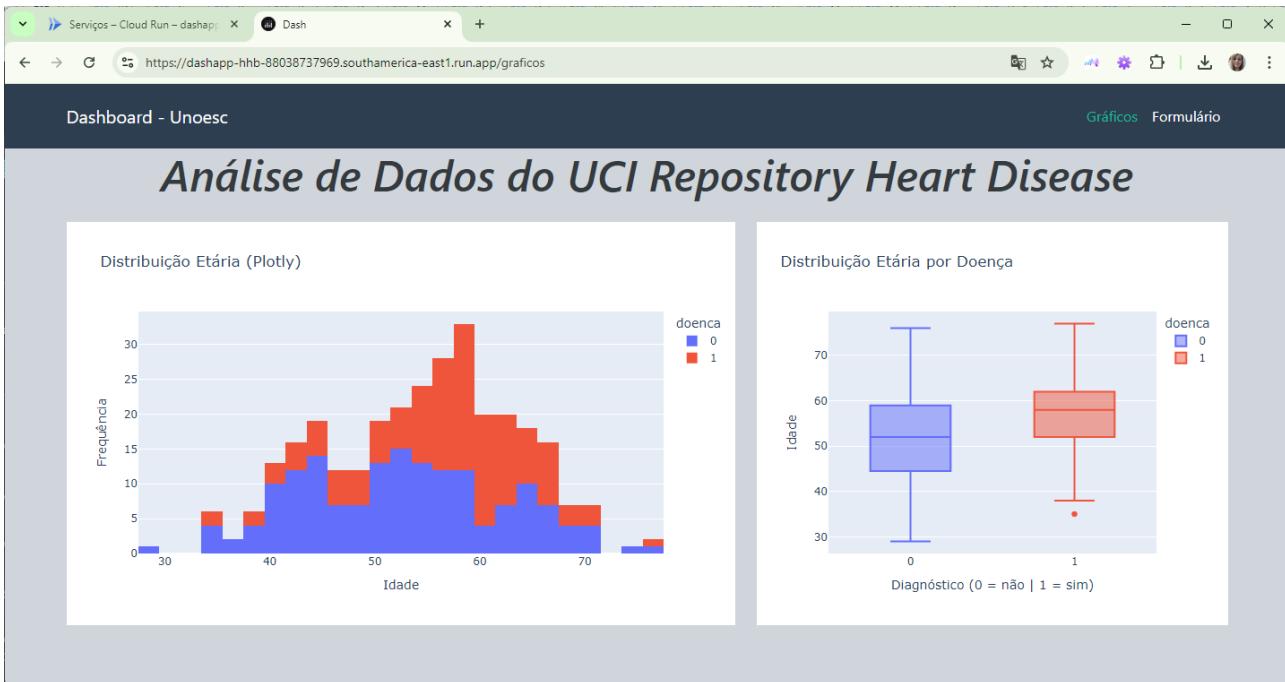


```
[21] europe-west3
[22] europe-west4
[23] europe-west6
[24] europe-west8
[25] europe-west9
[26] me-central1
[27] me-central12
[28] me-west1
[29] northamerica-northeast1
[30] northamerica-northeast2
[31] northamerica-south1
[32] southamerica-east1
[33] southamerica-west1
[34] us-central1
[35] us-east1
[36] us-east4
[37] us-east5
[38] us-south1
[39] us-west1
[40] us-west2
[41] us-west3
[42] us-west4
[43] cancel
Please enter numeric choice or text value (must exactly match list item): 32
To make this the default region, run `gcloud config set run/region southamerica-east1`.
Deploying container to Cloud Run service [dashapp-hdh] in project [dashapp-hdb] region [southamerica-east1]
OK Deploying... Done.
OK Creating Revision...
OK Routing traffic...
OK Setting IAM Policy...
Done.
Service [dashapp-hdh] revision [dashapp-hdh-00005-52g] has been deployed and is serving 100 percent of traffic.
Service URL: https://dashapp-hdh-416128962446.southamerica-east1.run.app
C:\programas\ml>
```

IMPORTANTE: Qualquer correção ou modificação feita no projeto irá exigir a construção de uma nova imagem com o comando
gcloud builds
e posterior republicação desta nova versão com o comando
gcloud run deploy

16.9 Teste do Projeto

Clique na URL fornecida segurando a tecla *Ctrl* para executar a aplicação na nuvem:



Painel do aplicativo:

The screenshot shows the Google Cloud Platform dashboard for the project 'dashapp-hhb'. The left sidebar lists resources: BigQuery, SQL, Compute Engine, Storage, Functions, and Cloud Run. The main area contains several cards: 'Informações do projeto' (Project ID: dashapp-hhb, Number: 88038737969, ID: dashapp-hhb), 'APIs' (Shows a graph of API requests over time with a peak at 15:15), 'Status do Google Cloud Platform' (All services are normal), 'Faturamento' (Estimated charges for the period 1 - 18 de ago. de 2025), 'Monitoring' (Link to Monitoring), and 'Error Reporting' (No error signals found).

Verificando a imagem em um bucket:

The screenshot shows the Google Cloud Storage browser interface. The left sidebar has sections for Cloud Storage, Informações gerais (Buckets, Monitoramento, Configurações), Storage Intelligence, Conjuntos de dados do..., Marketplace, and Notas de lançamento. The main area shows a table of buckets under the 'Buckets' tab. There is one bucket listed: 'dashapp-hhb_cloudbuild' created on 18 de ago. de 2025 at 14:33:21, located in Multi-region (us), with Standard storage class. A sidebar on the right provides links to help with Cloud Storage usage, such as 'Como receber informações do bucket', 'Como fazer upload de objetos', and 'Como fazer download de objetos'.

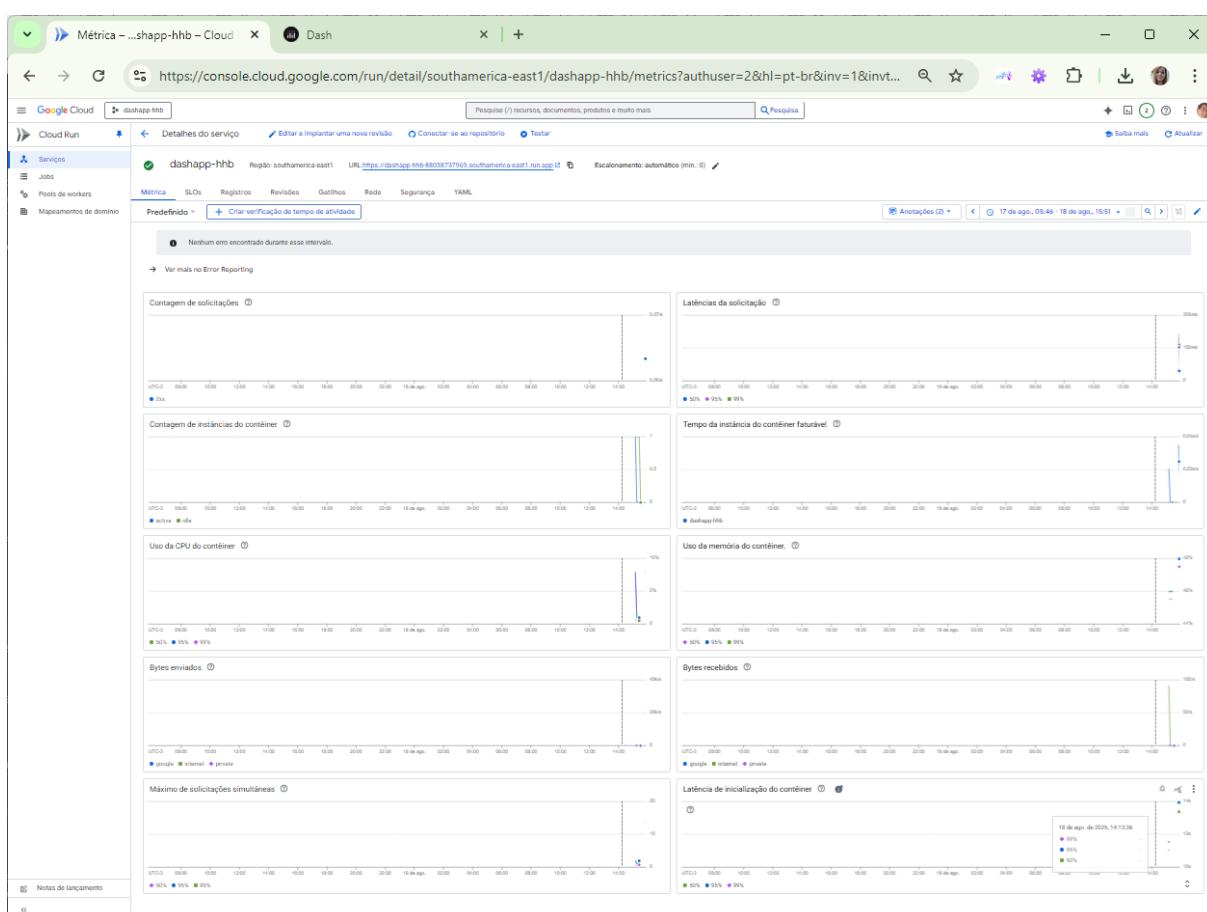
Item 'Cloud run', para executar a aplicação:

The screenshot shows the Google Cloud Storage interface. On the left, the navigation menu is expanded to show 'Cloud Storage' under 'Cloud Storage'. Below it, there are sections for 'Serviços', 'Jobs', 'Pools de workers', and 'Mapeamentos de domínio'. A modal window is open over the main content area, listing these options. The main table displays a single bucket entry: 'dashapp-hhb_cloudbuild' was created on August 18, 2025, at 14:33:21, is located in 'Multi-region' (us), and has a 'Standard' storage class.

Serviços:

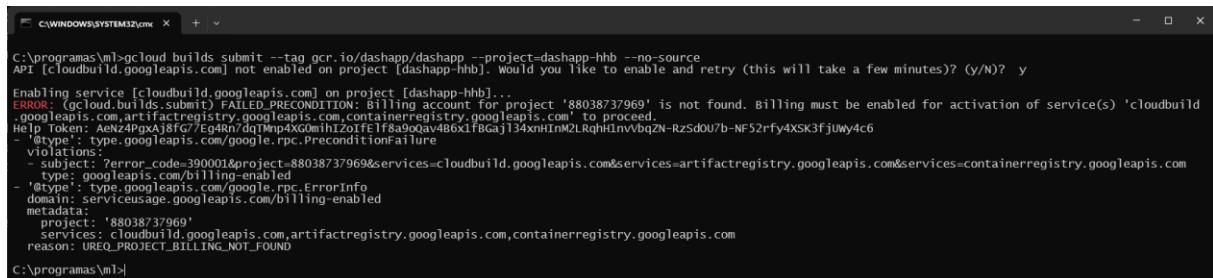
The screenshot shows the Google Cloud Run interface. The left sidebar lists 'Serviços', 'Jobs', 'Pools de workers', and 'Mapeamentos de domínio'. The main area displays a table for services. One service, 'dashapp-hhb', is listed with the following details: Name is 'dashapp-hhb', Type is 'Contêiner', Region is 'southamerica-east1', Authentication is 'Acesso público', and Last Implementation was 'há 45 minutos' by 'hercules.debiasi@gmail.com'. A note says: 'Cada serviço expõe um endpoint exclusivo e escalona automaticamente a infraestrutura para processar as solicitações recebidas. Implante uma imagem de contêiner, um código-fonte ou uma função para criar um serviço.'

Monitoramento:



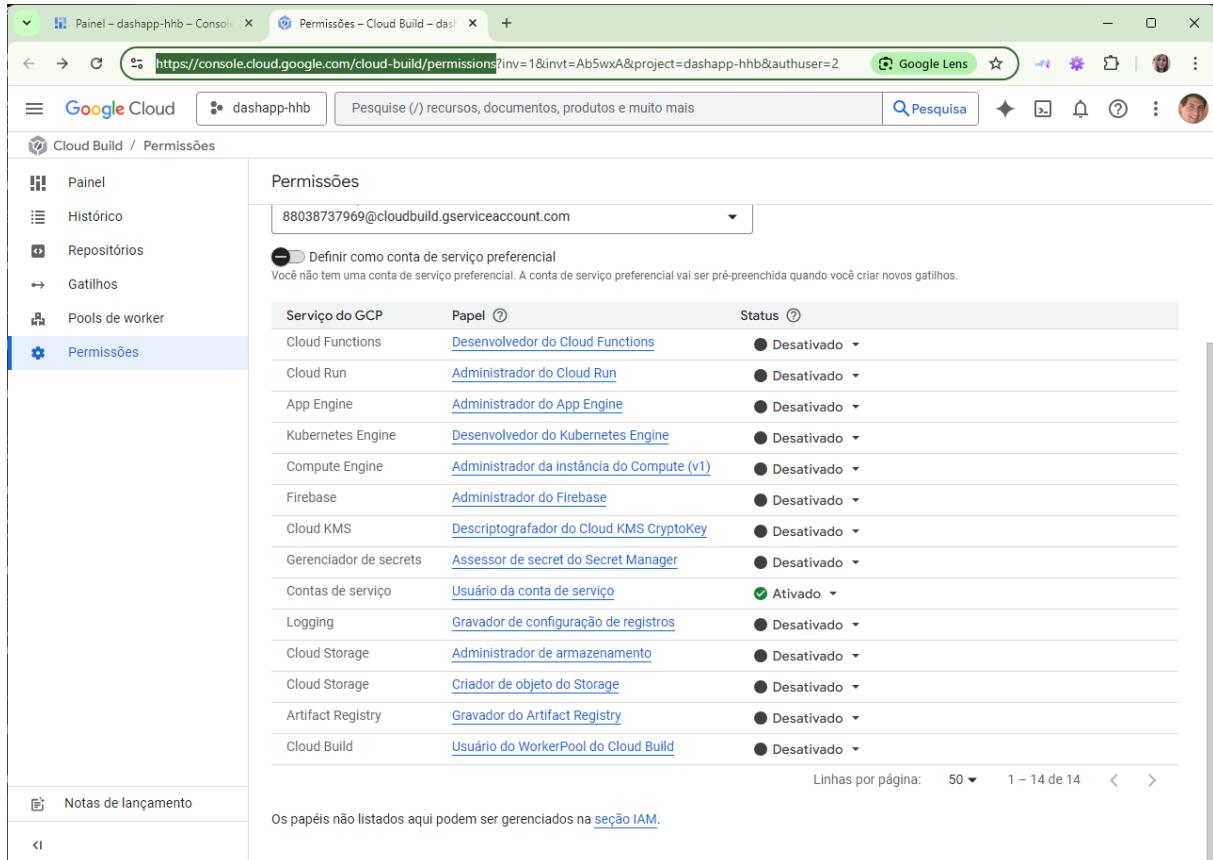
16.10 Solução de Problemas

Se a mensagem de erro abaixo aparecer é porque o faturamento precisa ser habilitado para o projeto.



```
C:\programas\ml>gcloud builds submit --tag gcr.io/dashapp/dashapp --project=dashapp-hhb --no-source
API [cloudbuild.googleapis.com] not enabled on project [dashapp-hhb]. Would you like to enable and retry (this will take a few minutes)? (y/N)? y
Enabling service [cloudbuild.googleapis.com] on project [dashapp-hhb]...
ERROR: (gcloud.builds.submit) FAILED_PRECONDITION: Billing account for project '88038737969' is not found. Billing must be enabled for activation of service(s) 'cloudbuild.googleapis.com.artifactregistry.googleapis.com,containerregistry.googleapis.com' to proceed.
Help Token: AenZ4PgxAj8fG7/Eg4RnjdqMn4XG0mih1Zof8a9oQa4B6x1fBGaJ134xhInM2LrqhInvbqZN-Rzsd0U7b-Nf52rfy4XSK3fjuWy4c6
- @type: type.googleapis.com/google.rpc.PreconditionFailure
  violations:
    - status: {error_code: 390001&project=88038737969&services=cloudbuild.googleapis.com&services=artifactregistry.googleapis.com&services=containerregistry.googleapis.com}
      type: googleapis.com/billing/enable
    - @type: type.googleapis.com/google.rpc.ErrorInfo
      domain: servicemanage.googleapis.com/billing-enabled
      metadata:
        project: '88038737969'
      reason: UREQ_PROJECT_BILLING_NOT_FOUND
```

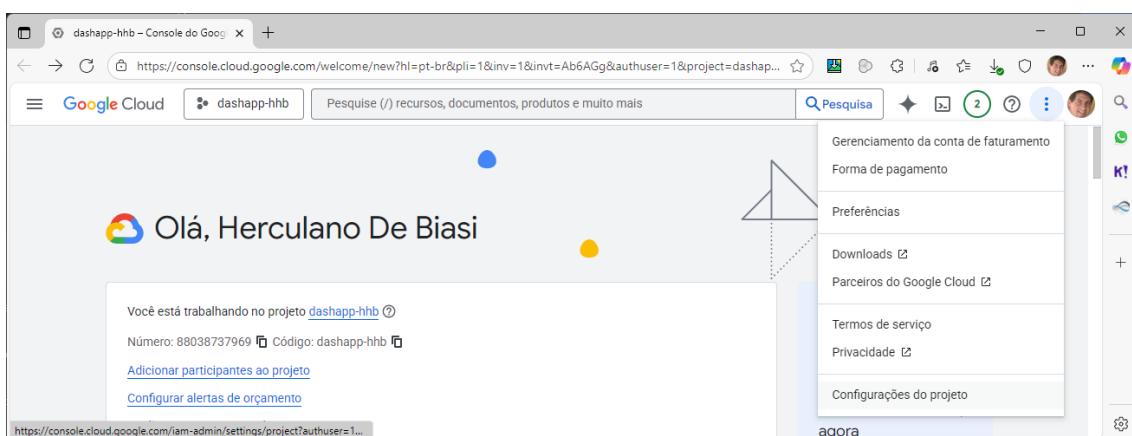
Em caso de erro de permissão entre na página de [permissões do Cloud Build](#) e ative a opção *Contas de serviço*:



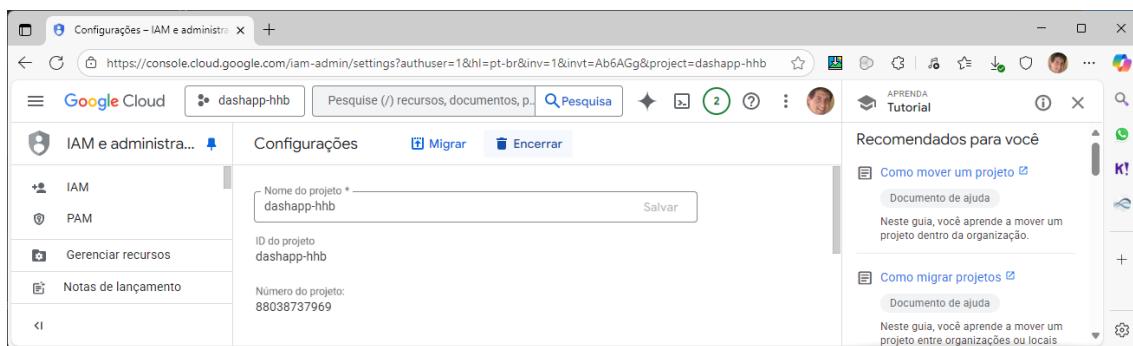
Serviço do GCP	Papel	Status
Cloud Functions	Desenvolvedor do Cloud Functions	Desativado
Cloud Run	Administrador do Cloud Run	Desativado
App Engine	Administrador do App Engine	Desativado
Kubernetes Engine	Desenvolvedor do Kubernetes Engine	Desativado
Compute Engine	Administrador da instância do Compute (V1)	Desativado
Firebase	Administrador do Firebase	Desativado
Cloud KMS	Descriptografador do Cloud KMS CryptoKey	Desativado
Gerenciador de secrets	Assessor de secret do Secret Manager	Desativado
Contas de serviço	Usuário da conta de serviço	Ativado
Logging	Gravador de configuração de registros	Desativado
Cloud Storage	Administrador de armazenamento	Desativado
Cloud Storage	Criador de objeto do Storage	Desativado
Artifact Registry	Gravador do Artifact Registry	Desativado
Cloud Build	Usuário do WorkerPool do Cloud Build	Desativado

16.11 Exclusão do Projeto

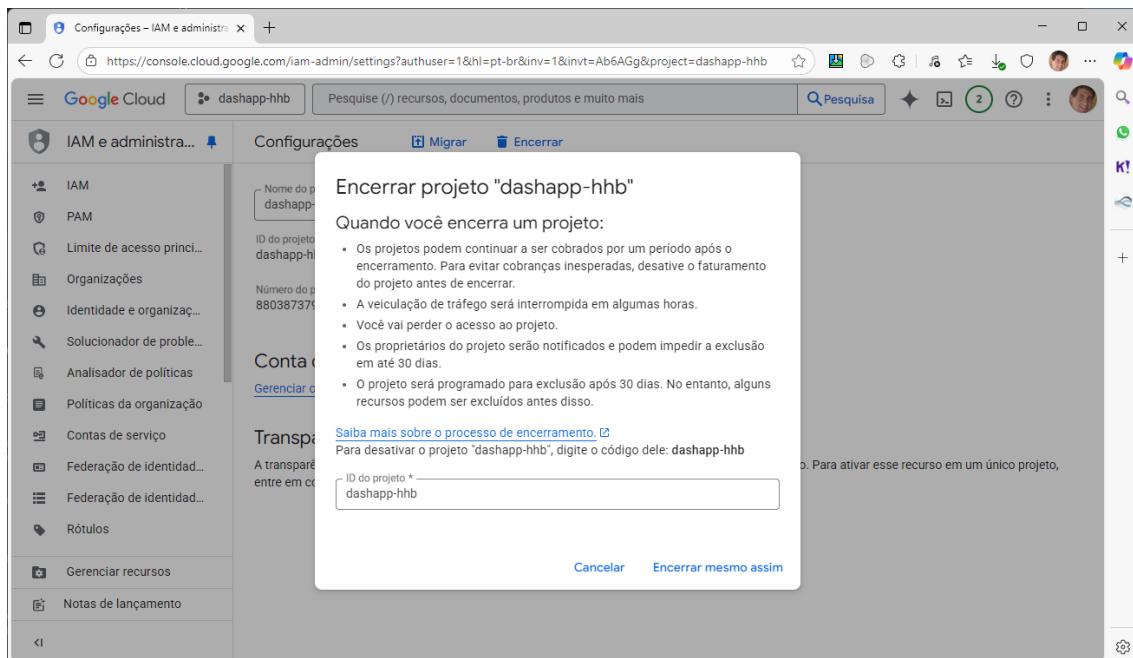
Após receber a nota do trabalho, pode-se excluir o projeto para evitar que o Google continue consumindo seus créditos ou realize cobranças. Caso deseje excluir o projeto, pode seguir esses passos: Primeiramente, clique no nome do projeto, o que abrirá uma janela contendo todos os seus projetos. Na parte superior direita desta janela, há três pontinhos. Ao clicar nesses pontos, será exibida uma lista de opções onde você pode selecionar '*Configurações do projeto*'.



Clique no botão 'Encerrar':



Insira o ID do projeto e clique em 'Encerrar mesmo assim':



Aviso:

